SELF-CONFIGURING AD-HOC NETWORKS FOR UNMANNED AERIAL SYSTEMS

A Thesis
Presented to
The Academic Faculty

by

Hans Claus Christmann

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Aerospace Engineering in the
Daniel Guggenheim School of Aerospace Engineering

Georgia Institute of Technology
May 2008

SELF-CONFIGURING AD-HOC NETWORKS FOR UNMANNED AERIAL SYSTEMS

Approved by:

Eric N. Johnson, Advisor
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Eric Feron
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Amy Pritchett
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: March, 31st, 2008

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

COTS	Commercial off the shelf.
DBP	Database Processor.
DHCP	Dynamic Host Configuration Protokol.
GNC	Guidance, Navigation, and Control.
GUST	Gerogia Tech UAV Simulation Tool.
HALE	High Altitude Long Endurance.
HITL	Hardware in the Loop.
HLA	High Level Architecture.
ISR	Intelligence, Surveillance, Reconossaince.
JAUS	Joint Architecture for Unmanned Systems.
LoS	Line of Sight.
MALE	Medium Altitude Long Endurance.
MANET	Mobile Ad-hoc Network.
MAV	Micro Aerial Vehicle.
MESH	IEEE 802.11s MESH Network.
PAN	Personal Area Network.
SAR	Search and Rescue.
SatCom	Sattelite Communication Link.
SIGINT	Signal Intelligence.
SITL	Software in the Loop.
UAS	Unmanned Aerial System(s).
UAV	Unmanned Aerial Vehicle.
UAVRF	(Georgia Tech) UAV Research Facility.
UGV	Unmanned Ground Vehicle.
VANET	Vehicular Ad-hoc Network.
WLAN	Wireless Local Area Network.

GLOSSARY

broadcast A braodcast is a transmission of a network packet to all nodes in range. Broadcasting is part of the triple unicasting (transmission to one destination), multicasting (transmission to several destinations), and broadcasting (transmission to all destinations in range)., p. 45.

download Download or downloading is used in reference to network traffic originating at a UAV and then send to a GCS., p. 32.

hop In a networking context, a hop describes the transmission of a packet from one node to a neighboring node. The notion hop can be used to quantify the number of necessary transmissions on a route or it can be used to describe the distance of nodes in a network topography map. In the latter, one hop is equivalent to an edge in a connected graph., p. 17.

latency Latency refers to the delay in the delivery of a network packet or an interpretation of the content thereof., p. 9.

network topology This refers to a certain network configuration and the related connection pattern. Even though the physical arrangement of the network nodes might be stationary, the topology could change - reflecting a different routing scheme for example., p. 6.

node A node is any participating entity in a network., p. 3.

ownship The ownship is the network node which could also be called the host to the referring item or process. In the context of software, for example, ownship refers to the network node that executes the particular piece of software under consideration., p. 44.

physical topology This refers to a certain physical arrangement of nodes and how they are located with respect to each other. The physical topology is the placement of the nodes on a map and a related connection pattern based upon which connections are physically possible. (In most figures this is indicated by lighter colors.), p. 14.

routing metric This refers to an associated cost of a ceratain decision on routing. A routing metric provides the means to minimize this associated cost in order to find the optimal solution - with respect to this metric., p. 9.

upload Upload or uploading is used in reference to network traffic generated by the human user and then send to the UAVs., p. 28.

user The term user refers to any entity that utilizes the UAS network. This could be a control station operator looking at a video feed or a UAV, utilizing a sensor capability on a different node., p. 13.

CHAPTER I

INTRODUCTION

Currently, there is ongoing research in the field of Mobile Ad-hoc Networks (MANET) for several different scenarios. A large interest is in applications for vehicular traffic scenarios, mobile phone systems, sensor networks, future combat systems, and low-cost networking ([2], [1], [5], [3], [4]).

Research has focused on topology-related challenges such as routing mechanisms or addressing systems ([36]), as well as security issues like traceability of radio communication or encryption ([14], [21]). In addition, there are very specific research interests such as the effects of directional antennas for MANETs ([46]) or optimized transmission techniques for minimal power consumption ([6]) or range optimization ([7]).

Most of these research topics aim either at a general approach to wireless networks in a broad setting (and so operate on a more abstract level) or focus on an extremely specific issue that bundles software and hardware challenges into one tailored problem.

Unmanned aerial vehicles (UAVs), and unmanned aerial systems (UAS) in general, need wireless systems to communicate. Current UAS are very flexible and allow for a wide spectrum of mission profiles by means of utilizing different UAVs, according to the requirements at hand. Each mission poses special needs and requirements on the internal and external UAS communication, and special mission scenarios calling for UAV swarms increase the complexity and require specialized communication solutions.

UAS have specific needs not provided by the general research, but are, on the other hand, too diversified to make much use of the narrowly focused developments; UAS form a sufficiently large research area for application of MANETs to be considered as an independent group with specialized needs worthy of tailored implementations of MANET principals. MANET research has not tackled a general approach to UAS, although some sources show specific applications involving UAVs, e.g. [9], [20] or [16].

This work presents some new aspects for the development of ad-hoc wireless networks for UAVs and UAS and focuses on their specialities and needs in the framework of this work.

1.1 Outline of this work

A general overview about MANET principles is given in Chp. 2 and major differences between different MANET branches are briefly outlined. The chapter also presents the results of a literature survey. Based on the insight gained by this overview of ideas and principals driving the different MANET types, Chp. 3 states requirements, targets and design specifications which are imposed on a MANET and which can be utilized as a performance metric for a successful operation in a UAV environment. The specifics of this evaluation environment are given in Chp. 4. Based on the given performance parameters and the evaluation scenario, the selection and decision making process leading to the proposed MANET framework is given in Chp. 5. Chp. 6 contains an overview of a first implementation possibility for the proposed MANET framework and leads to Chp. 7, which states conclusions and remarks related to the presented efforts as well as future work.

CHAPTER II

BACKGROUND

Based on general ideas presented previously about ad-hoc networks ([32]), and problems pointed out in an initial work on implementation ([20]), the starting point for this work is the vast field of mobile ad-hoc networks (MANET).

2.1 Ad-hoc Networks and Mobile Ad-Hoc Networks

Ad-hoc networks rely solely on their participating nodes to establish and maintain a networking capability. No external network infrastructure, such as network cables, landline backbones, or radio transmission towers is necessary. The network relies only on the capabilities of the participating nodes, the hardware they provide, and a common protocol.

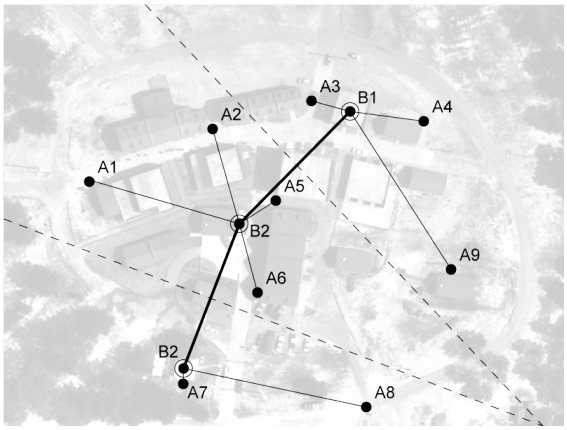
Mobile Ad-hoc Networks, or MANETs, are ad-hoc networks which allow motion amongst the participating nodes and hence have to deal with another layer of complexity. This increase in complexity generally results from the need to update topology information such as a list of neighbors and, depending on the particulars of the specific system, the distribution of that information.

2.1.1 Current Day Examples

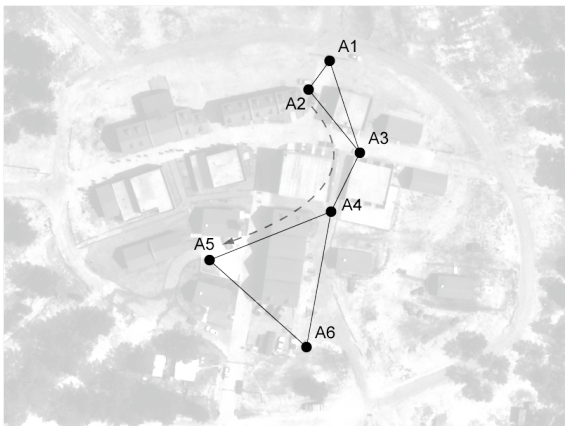
Examples that initially come to mind are the cellular network, the ad-hoc networking mode available to modern-day, WiFi equipped notebooks, or the Bluetooth Personal Area Networks (PAN). Unfortunately, all lack ceratin features which will be stated necessary for a MANET in a UAS environment later.

The cellular network, Fig. 1(a), relies on transmission stations to form the cells that make up the network; that is, it needs external infrastructure and, hence, disqualifies as a MANET.

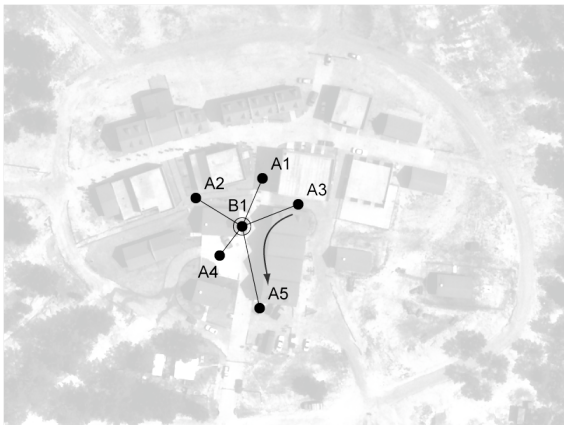
The ad-hoc mode specified by the IEEE 802.11b standard allows for participating nodes to form an impromptu wireless network, Fig. 1(b). However, the connections in that network



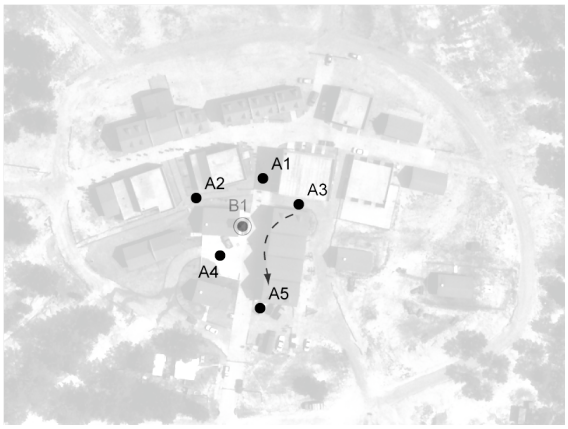
(a) Cell towers are a necessary infrastructure for mobile phone networks.



(b) 802.11b Ad-Hoc mode does not provide any routing capabilities. Nodes can only connect if they are in range of each other.



(c) A Bluetooth PAN provides routing and does not require infrastructure...



(d) ...but its star topology collapses if the master fails or is unavailable.

Figure 1: Examples of mobile networks that might come up initially prove to have some flaws when inspected from a MANET point of view. None of the given examples qualify as a MANET.

are peer-to-peer based, and such no routing functionality is provided inside the network. Network nodes can only communicate if they are within range of each other. However, there are third party extensions available to provide this functionality, but these are then considered as a separate protocol.

The Bluetooth PAN, an implementation of a Piconet([45]), can only be established between eight active nodes at a time, one master and up to seven slaves, Fig. 1(c). Even though the established star topology allows for slaves to communicate with each other by utilizing a routing functionality provided by the master, the network collapses in case of a failure of the master, Fig. 1(d). Hence, the master node could be considered a necessary infrastructure.

These examples outline three major problems in MANETs: independence of infrastructure, routing, and robustness. All of these will be seen again during the design stage of any new MANET protocol.

Currently, there is only one larger MANET implementation - or plan thereof - known to the author: the MESH network proposed in IEEE 802.11s and a first implementation in the devices of "One Laptop Per Child".

Other examples of MANETs in use are implementations of the ad-hoc routing protocols AODV ([28]) and DSR ([17]) on some patchable wireless router boxes, for example Kernel AODV [25], AODV-UU [39], and DSR-UU [40]. These efforts bring working MANETs into life but can only be classified as research or enthusiasts' efforts without a large user base.

2.2 Literature Survey

Based on protocol publications, the underlying baseline assumptions for the scenarios, and their effects on the choice of a protocol, a rough overview of the methods and techniques available as well as their general benefits, drawbacks and preferred environments was obtained. Although the number of protocols proposed as of now is already huge ([42]), the literature survey not only showed that there are only a few major categories within the protocols, but that there really are basic performance differences, shown, for example, in [30].

To explain the diversity of MANETs and to further motivate the use of an already existing MANET protocol as a base foundation, a rough classification is given below in order to indicate the enormous possibilities the different protocols already in existence offer.

2.2.1 Three Basic Principles of Routing Protocols

Routing refers to the process of how a path through the network topology is found, maintained, and used. The commutation method should pick a beneficial principal based on its needs and reconfigure itself appropriately.

Routing is a general challenge in a MANET. Based on the main routing principle, in general the protocols can be classified into three major groups: flooding protocols, reactive protocols and proactive protocols.

Flooding Protocols mainly follow the simple principle to transmit a packet to every node in the network in order to ensure it reaches the addressee of that specific packet. A dedicated route from source to destination is non-existent.

Proactive Protocols constantly maintain a current topology map or routing table with up-to-date information and generate the route to the destination thereof. The source nodes (in general) have at least a partial route to the destination before they start transmitting a packet.

Reactive Protocols operate contrary to proactive protocols and hence do not maintain a map or look-up table for routes. Instead, every time a message is sent it is necessary to query a route to the destination on the fly. Here, the source node only starts the routing process after it is tasked to send a message.

Several protocols have been developed in each group based on these three different principal routing mechanisms, but a strict classification is not always possible since protocols might use different routing methods for different purposes in different cases. This becomes obvious if the details and/or possible modifications to the three basic principals are further investigated.

2.2.1.1 Flooding in MANETs

As the terminology suggests, flooding is not a very subtle approach to routing. The related overhead is large and the network has to deal with related problems such as congestion. On the other hand, flooding does not depend on any knowledge of the network and as such provides benefits which make it a good choice for fall-back mechanisms. Interim steps in MANETs might use steps that utilize smart versions of flooding as a route discovery procedure and then use other means to perform route maintenance. One example for this would be an algorithm that floods the network and keeps track of the performed hops in each packet. The destination node would send an acknowledgement to the source by using the reverse route of the first packet that reached it and would thereby establish a new known route. This idea is used in part by the DREAM protocol, described in [35].

2.2.1.2 Proactive Principles in MANETs

The initial motivation in proactive protocols is to provide a look-up table for all packet destinations. This idea has been implemented in countless different proposed protocols, and many different approaches have been taken. The main problem is how to store the information related to the network topology and how to transform that into a hop-by-hop route for each packet and destination. Especially in this category, the nature of the underlying network plays an important role. Sensor networks (assuming stationary sensors), for example, often utilize proactive principles since updates are rarely necessary once the table has been built.

Depending on the protocol, different systems to obtain a routing table have been proposed. Three subgroups of proactive protocol principles are declared in order to classify MANET protocols better:

Address Based Systems are one way to find a route to the destination. This system is comparable to the system used in regular TCP/IP ethernet networks. Each node has a certain knowledge of its neighbors and knows a fall back path for any destination that is not known internally. In TCP/IP networks, for example, the packets are passed to the default gateway. Whether address based systems are strictly proactive might be

debatable since a hop-by-hop route is not necessarily known to the sender. However, each node knows *a-priori* to the send event how to handle the packet and so, for this work, they are considered proactive. The GLS protocol ([23]), for example, utilizes an address based system of location servers that maintain a route to the nodes they supervise. By utilizing these location servers, every destination is reachable by every source.

Geographical Location Aided Protocols utilize a knowledge of the geographic position of the networking nodes in order to find a route to a destination. How this location information is obtained is protocol dependant. Examples for this classification can be found in the ALARM [8], LAR [19], or DREAM [35] protocols.

Exploring Systems provide an algorithm that creates an internal routing table by means of a search algorithm. An example for a protocol in this group would be DSDV ([29]). Another simplistic example for such an algorithm would be the previously mentioned recovery procedure of the DREAM protocol.

As it is already obvious, existing MANET protocols are hard to classify and most often belong to more than one group. The protocol DREAM, for example, was already mentioned twice.

2.2.1.3 Reactive Principles in MANETs

Reactive protocols avoid the overhead in network traffic by not keeping a topology map or routing table. Instead of that, reactive protocols have to perform a route finding algorithm every time they want to send to a new destination. AODV ([28]) is one example of a reactive system.

However, the vagueness of this classification is easily visible, since simply storing an already discovered route, for example in order to send the next packet in an FTP sequence, could technically be called a reactive method. For this work, any protocol that at any point in time does not have a complete routing table is considered at least partially reactive.

2.3 Routing Metrics

In addition to finding a physically feasible route, a proper route must be chosen in case of multiple physically possible routes.

In this work the term routing metrics is used to describe these cost function comparable features. Finding a route, then, is comparable to minimizing the cost over all physically possible routes.

Several different metrics have been proposed so far to account for different application scenarios for MANETs.

Shortest Path Algorithms are surely the most common algorithms used. This metric minimizes the number of hops for each route. This is comparable to a shortest path algorithm in a traditional graph with all edge weights being equal and also has the benefit of being a good indicator for the latency introduced by this route. Since every hop introduces processing time on the packet, the latency can be estimated as being proportional to the number of hops. This assumes a heterogeneous network in the sense that all packet processing is done equally fast at all nodes.

Throughput Optimization is a more advanced metric and would take the available bandwidth on each node into account. Having an algorithm that includes this in its route selection routine helps to balance the traffic load over the network and optimizes the general network throughput.

Energy Aware Routing is a feature commonly found in sensor network protocols. These routing schemes try to prolong the lifetime of a remote sensor by means of reducing power requirements for transmission. This is done, for example, by synchronizing transmit and receive times at nodes (for example only daily between certain hours) or by minimizing the necessary total Rx/Tx power over a route if the nodes have the capability of adjusting their Tx power. In such a scenario, a route with more, but shorter, hops can be more energy efficient than a single long range transmission[1].

[1]Some protocols even explore the usage of directional antennas in that setting.

Network Security Concerns are a relatively new topic in discussing MANETs. Since MANETs do not require infrastructure, they are predestined to be used in an exploration or military setting. The latter always raises the question of signal intelligence (SIGINT) and its derivatives. Hence, some protocols explore the traceability of packet flow inside the network or the vulnerability to man-in-the-middle attacks where a hostile node is situated inside the coverage area of a friendly network. ANODR ([21]) is one example for a security aware MANET protocol. Other approaches could look at minimizing the electronic footprint of a network and are related to the energy aware approaches.

2.4 Other General Challenges in MANETs

In general, sending packets through a network is a time critical process and, especially for sensor networks, a common perception of time is necessary. In classical TCP/IP networks the network time protocol (NTP) is a *de facto* standard and is also widely used to synchronize nodes in the internet. Unfortunately, most classical approaches to time synchronization do not work in MANETs ([33]), and so other means have to be pursued.

Another general problem is the interconnection of MANETs with classical networks, for example wired TCP/IP networks. TCP tends not to perform well in ad-hoc environments ([24]) and so a transition point most likely has to be defined. This point would handle the conversion from the exterior protocol, e.g. TCP, to the MANET protocol at hand. The challenge is to decide how and where to implement this capability and how to resolve the most likely issues such as time-outs.

Further challenges with MANETs exist in addition to these. Some will be outlined in Chp. 3, where the direct influences on the development of the proposed MANET protocol are also stated.

2.5 Current State of Research

Since the introduction of the IEEE 802.11 standards, there has been interest in MANET-related challenges such as no infrastructure and the general problem of routing. There are a few MANET protocols which stand out, either due to their uniqueness and principal or

due to their status as a reference protocol. DSR, AODV, TORA, and AODV are some of these reference protocols[2] ([17], [28], [27], [28]).

As pointed out initially, so far there is no large implementation of MANETs in real applications, possibly due to the lack of a common IEEE standard. Since IEEE 802.11s is going to introduce an industry standard for mesh networks, this may change.

So far, MANET research is mainly performed by proposing a (new) protocol, simulating it in a network simulator or a small hardware testbed, and comparing the performance to other (reference) MANET protocols. Research splits up into the development of new protocols, adopting current standards for use in a MANET; and creating better simulations.

[2]This might also be due to the fact that these protocols are easily available in the tools used for MANET development.

CHAPTER III

EVALUATION METRICS

The goal of this work is to develop a truly tailored protocol of a self-configuring communication network for an UAV environment. This protocol will benefit from implementing certain features proposed in different MANET protocols and avoid drawbacks resulting from inappropriate features of protocols not being developed for a comparable environment. This goal has guided the approach to the problem, i.e., the outline of this work. A set of requirements which should be matched was created as a common baseline for all future work. This set is used as a metric to gauge performance, decide between options, or, in general, to guide further progress.

These requirements and the evaluation against them are responsible for the adaptation process of the communication method for an UAS environment. Hence, the requirements are based on the needs of the UAS community in order to optimize the results towards the intended use. These requirements are formulated in this chapter .

3.1 Self-configuration

Since UAVs have an autonomous nature, the communication mechanism should be self-configuring. This means that, similar to the difference between manual static assignment of IP addresses and the usage of a dynamic host configuration protocol (DHCP) in TCP/IP computer networks, the communication network for a UAS environment should require minimal human interaction in order to operate.

The participating UAVs and control station units should go through an initial configuration process which should be dependent only on the hardware set-up of the node at hand. in this process the physically connected subunits related to networking and their corresponding parameters are configured. This could be, for example, the type of communication link (for example WiFi, wireless serial, SatCom), a rough range model for these

systems, the expected power consumption or other specifications such as available or expected bandwidth and latency. After this initial configuration, every participating network node should be able to perform any further necessary steps on its own, without a mandatory user interaction or extended low-level configurations.

This should also explicitly include the problem of *instantiation* of vehicles, the *routing* of packets and the possibility to *expand* the network.

3.1.1 Instantiation

Instantiation refers to the action of adding a new node into the life network, which should be achieved by a simple process that does not require a (manual) reconfiguration of all other network nodes. Once a network node has been initially configured, the instantiation should be as easy as possible.

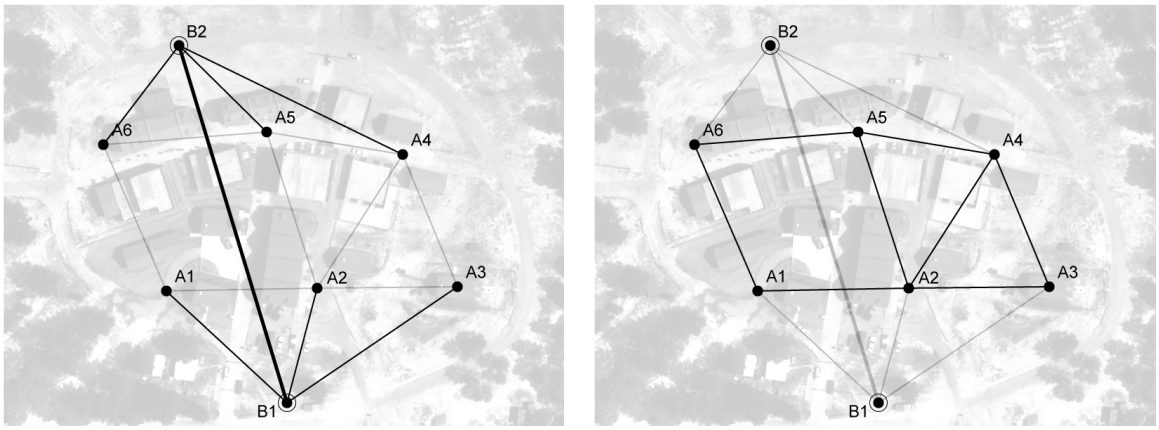
One possible scenario would be that a node simply would be powered up and automatically connected to the network if any member of it is in range. A second option would be that the vehicle, if in range of any network member, would request and require permission from a control station operator to join the network.

Instantiation should also be used to announce vehicle features to the network. This could include announcing special communication features (like a SatCom capability), immediately relevant for the functioning of the communication mechanism, or the announcement of certain task related capabilities (like the available sensor or actuator capabilities).

3.1.2 Routing in UAS MANETs

Routing not only refers to the process of finding a possible route, but also to the process of finding a beneficial one. Dependent on the mission and the current task at hand, an appropriate metric (see Sec. 2.3) has to be applied. As shown in Fig. 2, several topology configurations might be feasible. If one node is, for example, streaming high amounts of data to several other nodes, the throughput optimized configuration (Fig. 3.1.2) could be beneficial whereas (even at the same time) the latency optimized configuration (Fig. 3.1.2) could be better for smaller amounts of data for only a single other node.

Routing is heavily influenced by the choice of MANET protocol and, hence, a major



(a) Backbone configuration for optimized throughput

(b) Latency optimized point-to-point configuration

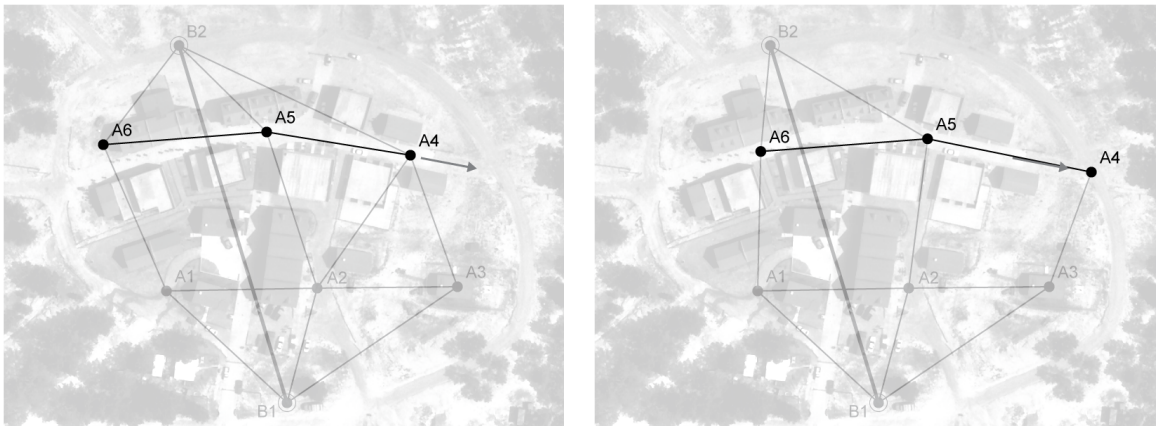
Figure 2: The desired communication method should be adaptive to the present requirements of the mission. Automatic switching of a routing metric, dependent on the type of mission in general or even dependent on the type of message, provides optimal performance with respect to the task at hand.

factor in deciding which protocol to choose as a development basis. As a matter of fact, as outlined in Sec. 2.2.1, the routing algorithm is the main distinction between most protocols.

3.1.3 Expandability of the Network

The network should be able to expand in a topological sense by means of initializing added nodes, as stated in Sec. 3.1.1. It should be possible to add and remove vehicles or control stations during runtime without an extensive workload to the human user. Simple log-on and log-off commands should be sufficient. The network should therefore not only not be limited *a priori* by a limitation on the amounts of its members, but the address system has to provide a sufficiently large address space to accommodate new nodes.

However, the network should also be expandable in a geographic sense and be able to be spread out over a larger area. Expandability also implies that the network should be able to make use of the participating units in order to increase network coverage. Hence, the network should not be solely based on any kind of fixed physical topology structure, but should allow for mission and scenario dependent configurations. Fig. 3 shows a change in node location, i.e., the geographical position of the vehicles change in order to maximize radio coverage area without changing the network topology. In this way, the network might



(a) Intended network expansion

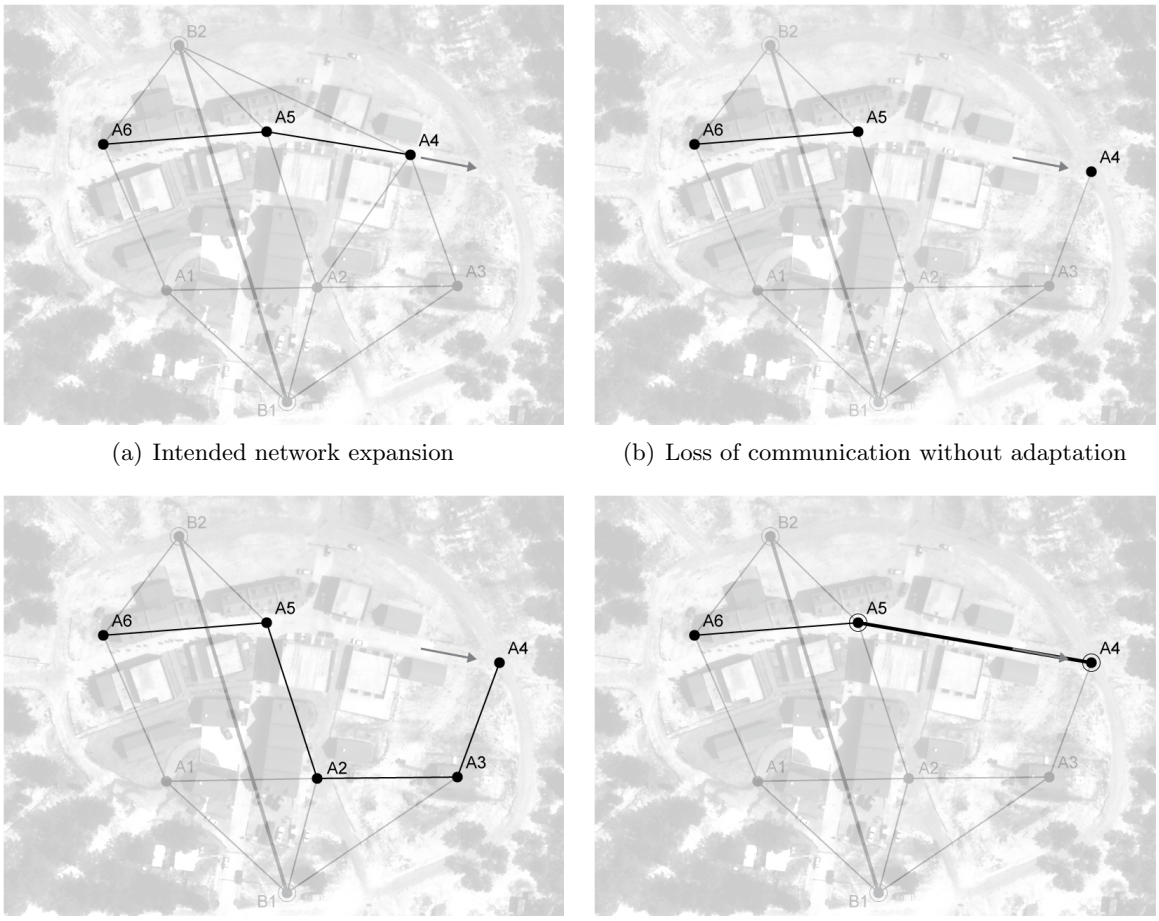
(b) Change in physical topology to accommodate this

Figure 3: The network should be able to handle on-the-fly changes in topology based on the physical distribution of network nodes. Since the range of each network node is limited a physical expansion of the network might require a physical relocation of the network nodes.

be able to provide certain demands, but this also requires the network participants to be flexible in their position, which implies processes that involve not only communication demands but also guidance, navigation and control processes.

However, since physical relocation is not always possible, the network has to utilize its routing capabilities to compensate for the change in physical topology. Since UAVs are again not only special in their requirements on a communication method, but also provide special features, a different solution than the classical re-routing might exist. Depending on the vehicles and control station units under consideration, communication hardware could be redundant and the vehicles might have a second, different, communication device in order to minimize errors due to similarities. Fig. 4 illustrates how this plays into routing: one of the nodes has relocated, Fig. 4(a), and would leave the radio range of a neighboring node, i.e. the communication link would be lost as shown in Fig. 4(b). Here, either re-routing, Fig. 4(c), or switching from a shorter range to a longer range link, Fig. 4(d), would keep or re-enable the link.

The way a protocol deals with these changes in topology are of the utmost interest since UAVs as a matter of fact are moving and the network covered area hence will physically expand and contract.



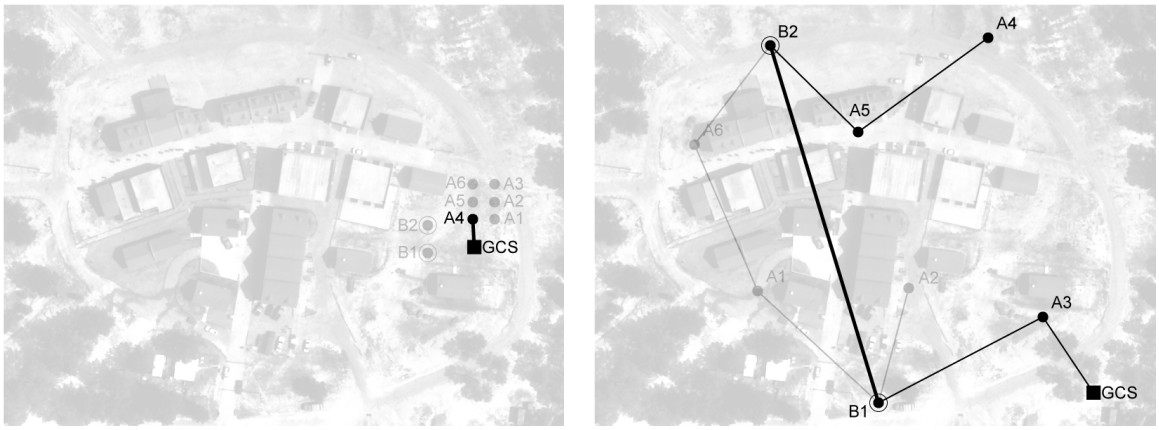
(a) Intended network expansion

(b) Loss of communication without adaptation

(c) Counter reaction based on rerouting

(d) Counter reaction based on changing the transmission devices

Figure 4: The network should also be able to handle the on-the-fly changes by a reconfiguration. The apparent way would be to rely on the routing algorithms, but since UAVs might have redundant communication devices, the switch to a device with different propagation properties could be a better solution.



(a) Direct connection, e.g. during maintenance work (b) Complex connection, e.g. during a live mission

Figure 5: Any user should be relieved from concerns about the actual network topology or low level configuration. The commands to retrieve or send data of any kind should be simple, self-explanatory and, most important of all, identical in every possible network setup.

3.2 Transparency

Transparency, in this context, mainly refers to the human operator at the control station and how much she or he needs to focus on reconfiguring the communication links. The network should not only be self-configuring and hence hide as much configuration as possible from the user in normal tasks, but also should implement the communication method in a way that a control station operator does not need to know if the UAV being addressed is directly connected to a control station or if the UAV is airborne and several hops away. Furthermore, users should not have to care about how their demand for information from a specific UAV or transmissions to UAVs fits into the general traffic situation on the network.

Transparency also could eliminate error sources since application level commands, such as retrieving or sending data, are implemented on a different and higher level than the actual communication layer. The user should not need any knowledge of how the UAV is reached. Both setups shown in Fig. 5, for example, should result in the same workload for the control station operator (with respect to communication tasks).

3.3 Mission Adaptivity

A UAS network must deal not only with the problems already mentioned, but also with a mission aspect: a high speed, close formation scenario poses different requirements on a communication method than a long term, high altitude surveillance scenario. The network has to be able to deal with low latency real-time requirements, high bandwidth requirements for video streaming, downloads of recorded data, or periods of enforced radio silence due to SIGINT concerns, for example.

In Sec. 3.1.2 the routing problem for bandwidth and latency optimization already has been introduced. The mission aspect is simply a different approach to the same idea: instead of a beneficial routing of the packets dependent on the kind of data, now a route is required which is beneficial for the mission at hand. The network has to allow for a metric for that as well.

This would mean, for example, that in a close formation flight the network shouldn't allow the allocation of bandwidth and transmission time to video streaming if this would result in drawbacks for higher priority traffic, such as collision avoidance messages. Since the real time transmission of position data has a higher priority in this scenario, it cannot be allowed to be degraded in order to allow for video streaming.

Another mission aspect would be the the incorporation of vehicle guidance and control into MANET concerns. In Fig. 3, the possibility of node movement is introduced as a method of maintaining connectivity. This action is governed by decisions outside the networking processes since guidance, control, and current trajectory constraints have to be checked. However, the routing agent proposed in Sec. 5.5.3 provides the base functionality to request exactly this action from the guidance, navigation and control processes of a UAV.

3.4 Compliance with Established Standards

Since this work develops a network architecture for an UAS (research) environment, the developed method should conform to or at least not contradict established industry standards. This ensures a high level of compatibility as well as comparability to other methods following the same standards.

3.4.1 The High Level Architecture

The high level architecture (HLA), defined in [34] as a guideline for multi-group simulations and activities and how to interact and exchange information, is considered to be a standard to meet. The HLA is considered by the author as a standard to meet in the simulation environment - which could and should include hardware in the loop (HITL) setups as well.

3.4.2 The Joint Architecture for Unmanned Systems

The JAUS Working Group, responsible for the joint architecture for unmanned systems, considers JAUS to be "[...] a component based, message-passing architecture that specifies data formats and methods of communication among computing nodes. It defines messages and component behaviors that are independent of technology, computer hardware, operator use, and vehicle platforms and isolated from mission." (Quoted from [15].)

Due to the nature of JAUS, full support can't be established by a MANET protocol, since JAUS also defines network messages and formats, especially for data packages (as compared to control packages, necessary for maintaining the MANET). JAUS implementation hence relies mostly on the formatting of messages whereas a MANET protocol is more focussed on the delivery of these messages.

3.5 Target Implementation Environment

Since the scope of UAS and UAVs are extremely diversified and applications and possible mission scenarios are vast, a representative implementation environment had to be found in order to perform a real evaluation and to obtain the necessary insight into real-world demands and requirements. The definition of a target implementation environment bridges the gap between theoretical requirements and evaluation in a real scenario by providing a metric for the feasibility of a future implementation. Keeping this in mind is not only necessary to be able to validate the reliability of the proposed MANET, but also provides requirements immediately resulting from the needs of a real environment and hence supports the idea of a tailored MANET protocol.

The Georgia Institute of Technology UAV Research Facility (UAVRF) was chosen for

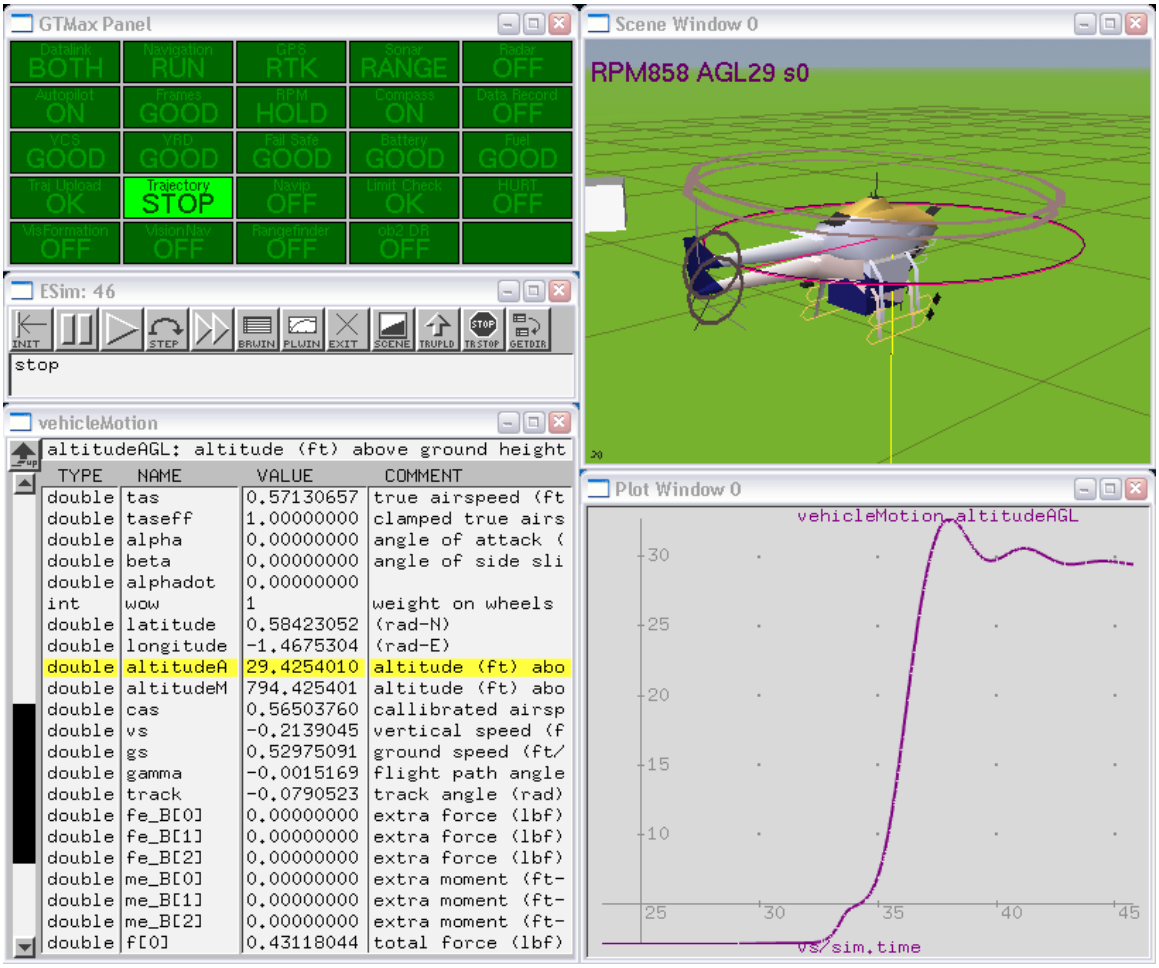


Figure 6: The GUST frontend ESim provides multiple views, full access to all variables used (including changing them) and realtime plotting capabilities for these variables. Built-in data recording capabilities allow for further analysis of vehicle behavior or any other related issue.

this purpose as representative of a UAS environment . The UAVRF possesses, builds, maintains, and operates several UAVs of different category, type, and capability. Due to the research orientation of the lab and the resulting necessity to be able to perform all kinds of missions, a wide spectrum of intelligence, surveillance, and reconnaissance (ISR) missions can be covered. At the UAVRF, a respectable amount of real-world requirements are known due to former and present interactions with non-research UAV users and operators. The UAVRF was therefore chosen to be the environment in which to eventually implement and validate the communication method under real-life conditions and whose immediate requirements on a communication architecture have to be met.

The UAVRF utilizes a combination of simulation, both SITL and HITL, and flight testing to develop and validate UAS features and capabilities. The UAVRF utilizes its own IT environment for that purpose, the Georgia Tech UAV Simulation Tool (GUST). GUST allows for modular extensions of the simulation environment and realtime changes of parameters due to an interaction between a custom database processor (DBP) and the simulation frontend ESim, shown in Fig. 6. GUST can be used to simulate the interaction between the control station and the actual UAVs. Since GUST contains the onboard code for the vehicles and is used to operate the control stations, it enables preparation for the first flight with newly developed features, for example the new communication method proposed.

In addition to the other requirements stated in Sec. 3, the proposed MANET communication method hence is to be adapted for use in the UAVRF. This requires not only adaptability to different hardware setups, i.e different UAVs, but also easy access to any kind of data on the UAVs in the network and the ability to easily record this data. A seamless integration into the already existing GUST environment is necessary to achieve this.

In order to create a versatile system, the communication mechanism has to be able to fit neatly into any kind of simulated environment at the UAVRF: the system should be able to handle any combination of real vehicles, simulated vehicles, one, none, or several ground stations, and the possibility that all software simulated entities are running on the same or different PC's or other computing devices.

Software in the UAVRF is written in C and C++, partially object oriented and uses DBP in order to create the necessary variables and their containing structures.

So far, the UAVRF uses wireless datalinks according to IEEE 802.11b, Freewave and Maxstream wireless serial connections, as well as wired serial and wired ethernet connections for internal communication.Hence, the communication mechanism will have to support at least these methods on the physical layer and should be able to use them in any combination or sequence.

CHAPTER IV

EVALUATION SCENARIOS

In order to create a test environment utilized for comparison and performance evaluation, it is important to describe not only the performance metrics but also the exact execution of that test. The performance metrics have been given in Chp. 3, and the execution of the test is described in this chapter. However, while a mostly generic set of scenarios for evaluation purposes facilitates a generally broader and less specific result, a real implementation environment in which the proposed MANET protocol could be validated is not only of interest but also contributes to the restrictions and requirements.

In order to evaluate the performance of a MANET in a UAS environment, this environment has to be specified and created. This is accomplished by presenting a number of reference scenarios. The goal of these scenarios is split between mimicking realistic UAS missions and simulating a realistic network utilization during the (simulated) mission execution.

4.1 The Actors: UAVs - And Why This Makes a Difference

Unmanned Aerial Vehicles compose the majority of the nodes in the discussed scenarios. This results in several different aspects that are seldom covered in MANET research.

4.1.1 Diversity Amongst the Nodes

At this point UAVs do not operate in swarms of dozens of vehicles of the same type. UAVs are still considered a unique asset providing special capabilities to their users. Based on this fact, a pictured UAS MANET will consist of several, physically different UAVs. This is true for the main areas of UAV applications:

Research Environments will normally gradually develop UAVs and hence end up with several different and differently equipped vehicles (if dedicated UAV swarm research

(a) The 160lbs GTMax Research UAV utilizes 802.11b/g, FreeWave Wireless Serial Links, and an Analog Video Transmitter

(b) The 2lbs AFI Hornet Micro utilizes a MaxStream Wireless Serial Link and an Analog Video Transmitter

Figure 7: The UAVRF operates several vehicles, fixed wing and helicopters, which are all able to perform the reference scenarios. Two rotary vehicles have been exemplarily picked to illustrate the diversity amongst UAVsand their networking features and capabilities.

is omitted). These will then either be operated in a single-vehicle-only fashion or in a heterogeneous network.

Military Environments tend to utilize UAVs also in a single-vehicle-only fashion (for example a squadron utilizing a MAV for local reconnaissance) or in a larger, more network centric oriented fashion (this would be given, for example, by a HALE UAV observing the global development, coordinating with several MALE UAVs as well as the local ground units and UGVs). The latter also comprises a heterogeneous network.

Civil Applications are mainly limited to SAR or border patrol applications. Examples of these applications show that UAVs in these scenarios are mainly used in a single-vehicle-only fashion[1].

The differences in the environments result in the requirement to introduce a diversity in the nodes when it comes to performance as well as capabilities. Current research mainly focuses on homogeneous networks.

[1]The US Department of Border Patrol plans to utilize a single Predator UAV at the US/Canadian border, for example (from "Unmanned Systems", December 2007). Missions involving UGVs at Ground Zero were also conducted in a single-vehicle-only fashion, even though the gathered information in both cases is shared afterwards.

In this work diversity is introduced by two types of vehicle roles, a supportive and an active one. Details on these roles and how they are utilized can be found in the description of the reference missions in Sec. 4.2. In general the role of the supportive UAVs could be understood as a carrier, an observing mothership, or, if the scenario would be made larger, as a refueling tanker.

Since the UAVRF operates as a target implementation environment, Fig. 7 pictures two UAVs operated by the UAVRF. The GTMax could be assumed to operate as a supportive UAV, the AFI Hornet Micro as an active UAV.

4.1.2 Realistic Motion Patterns

The necessity to replicate a realistic scenario has been realized in the field of MANET research a long time ago. Hence, different motion patterns have been developed to account for different behaviors of different networks.

Random Motion obviously is a first implementation. It is easy to create and confronts MANET networks with all kinds of network topology related problems due to motion. This motion pattern is used, for example, to replicate the movement of people in a mall, on a huge plaza, or during other big venue events, like concerts. Parameters are boundaries for velocity of the nodes, pause or halt times, behavior when reaching the boundaries of the simulated area, or areas of congestion (e.g. the food court).

Random Waypoint Models evolved from random motion models and mimic more natural motion consisting from stretches of linear motion, turns, and stops. This was done in order to avoid a Brown's motion-like behavior. In general these models generate a higher amount of directed motion, i.e. nodes do not tend to vibrate more or less stationary in a certain area.

Manhattan Scenarios try to mimic mainly vehicular motion, that is a random motion bound to certain passages (streets). These scenarios work well for vehicular ad-hoc networks (VANET) and replicate the problem of a more coordinated motion.

A common denominator among all these motion patterns is, that they all make sense for a *single node* and that the motion of each node is more or less independent of the motion of other nodes. In a UAS, unfortunately, this separation principle tends not to be true. In the mentioned examples, the whole heterogeneous network is utilized as a single entity in order to accomplish a certain task. This assumes the military example to be of a certain maximum size. Once the network grows too large, obviously several independent tasks can be performed by independent subgroups of a network. This leads to the fact that in order to create a reference (mid size) UAS scenario the whole scenario has to be coordinated and planned. The motion patterns of every single UAV has to happen in consistence with the motion of the others and the general task of the mission.

When it comes to representative motion patterns one feature that has a high impact is whether a UAV has the ability to hover or not. In order to maintain a broad base for possible real missions, the reference mission try to not be restrictive when it comes to that. If there are limitations, the mission will spell that out.

4.1.3 Representative Network Traffic

The goal of a truly tailored protocol for an UAS environment also calls for a realistic representation of the expected network traffic. This requirement results in two different factors.

First, the amount of data, the volume of traffic, has to be realistic. The digital communication in a UAS MANET consists of several different parts: protocol related traffic, UAV control related traffic, and mission related traffic.

Protocol Traffic means packets that have to be sent in order to maintain the network functionality. This includes messages exchanged during a node's instantiation as well as messages related to routing, like a possible receive acknowledgement or other low-level network functions.

UAV Control Traffic represents the traffic from control station units to the UAVs in order to set a flight plan, trigger a certain action at the UAV or perform any other kind of control over the nodes. Another part also contains the information the network is

providing to its operators in order to establish a general overview about the network status. This contains, for example, all kinds of health status or navigation related information, such as position, velocity, or current fuel consumption of a node. The information in this section in general provides situation awareness for the user.

Mission Related Traffic can be considered the payload of network traffic. This section covers sensor data streams or single critical sensor measurements. The mission related traffic either comprises the reason for the execution of the mission (SAR or reconnaissance missions) or is necessary for mission critical decisions (for example target confirmation procedures).

Some sensors provide analog data streams, however, e.g. analog video cameras. Since relaying this data in analog form is normally avoided due to signal quality concerns, UAVs carrying analog sensors normally operate within a single-hop distance from either the final destination of that data stream (most probably a control station operator) or an appropriate A/D converter. In both cases the analog transmission part in this work is considered negligible, either due to the fact that it is completely separated from the rest of the communication or due to the fact that it could be substituted by a digital data stream, originating at the node which is carrying the A/D converter.

Second, besides a representative traffic volume, also the direction of the traffic is of importance. In recent MANET research there are mainly two types of traffic distributions present. One is a random distribution of traffic source and destination, combined with a small traffic volume per transmission. The other main representation gives a limited number of connections which transfer packets over a prolonged period of time, hence creating a higher traffic volume per transmission and a more sparse distribution of traffic routes.

Even though both models provide certain focus areas for MANETs (the first on route detection and latency, the second on route maintenance and throughput), none of them is suitable for mimicking realistic UAS network traffic. This is mainly due to the fact that main parts of the traffic volume in a UAS network are very directed. Traffic normally is directed towards the users of the network, which is mainly equivalent with the control

station node(s) here. This is due to the nature of the UAS network, whose primary task most often is to gather intelligence, i.e. sensor data, and provide it to the human operator (at the control station or behind any other gateway node). This results in challenges due to congestion and contention in the area close to the control station, where area means the nodes in a one or two hop neighborhood of the applicable network topology.

Hence, comparable to the motion part, the network traffic part has to be tailored and created as a whole, adapted and matched to the actual mission and motion situation.

4.1.4 Representative Communication Hardware

Along with representative traffic comes the use and simulation of representative communication hardware. In order to stay representative, several different communication devices have to be considered. Unfortunately, especially in a military environment, proprietary or highly specialized communication links and Rx/Tx power specifications outside the legal civil limits are common. The availability of SatCom, for example, is realistic for a military HALE UAV, however, satellite communication poses extra challenges[2] and hence is ignored for this work. Following a similar reasoning, the list of representative communication units is limited to COTS devices.

4.2 The Scene: Reference Scenarios

Based on these special considerations for UAVs, a scenario has to be created. A scenario describes the combination of a UAS mission and the corresponding network activities, i.e. the network traffic. These two parts play into all the fields found to be relevant. The creation of the scenarios was driven by the idea to replicate all essential aspects for MANET development in at least one of the scenarios and at the same time create missions that contain all (or at least very many) aspects of realistic UAS missions.

[2]Due to the high latency and the imbalance of up- and downlink bandwidth, special ad-hoc protocols have been developed and simulated for satellite based communication (see for example the satellite section in [13]).

4.2.1 Network Traffic in the Missions

As outlined in Sec. 4.1.3, there are certain parts of the network traffic which are more or less independent of the mission at hand, and these are mainly the protocol and control traffic. The protocol traffic obviously does not need to be artificially created in a scenario, as it is introduced into the scenario directly by the MANET protocol. Hence, only the control traffic needs to be specified beforehand.

In order to obtain representative data, all scenarios copy the control traffic situation of the UAVRF. In this work, it is assumed that all mission planning has been done offline and necessary flight plans, trajectories, waypoints, etc., have been transferred a priori. Hence, in all scenarios, no traffic is generated at the control station and uploaded to the vehicles. On the downlink side every mobile node generates two messages during the whole mission execution time. One is a 68bytes message, sent at 10Hz, containing an essential set of data for situational awareness. The second message is a 184bytes message, sent at 1Hz, providing extended status information of the vehicle. As an initial requirement, both messages have to be at least sent to the control station in order to meet the requirement that a control station operator would want access to all the status and health information in order to obtain and maintain situation awareness. In order to mimic the mission data, each (simulated) sensor has been assigned a corresponding traffic volume. The occurrence of this traffic is scheduled according to the reference missions stated below.

Summing this section up, the total network load comprises of the following parts:

- Message 1: 68bytes at 10Hz, originating at a UAV.
- Message 2: 184bytes at 1Hz, origination at a UAV.
- Medium Bandwidth Data Stream: 300kbit/s
- High Bandwidth Data Stream: 1Mbit/s
- Single Event Sensor Data: 2MB

The data for the data streams and the single event sensor data was chosen to mimic medium and high quality digital video streams as well and a digital still picture, respectively.

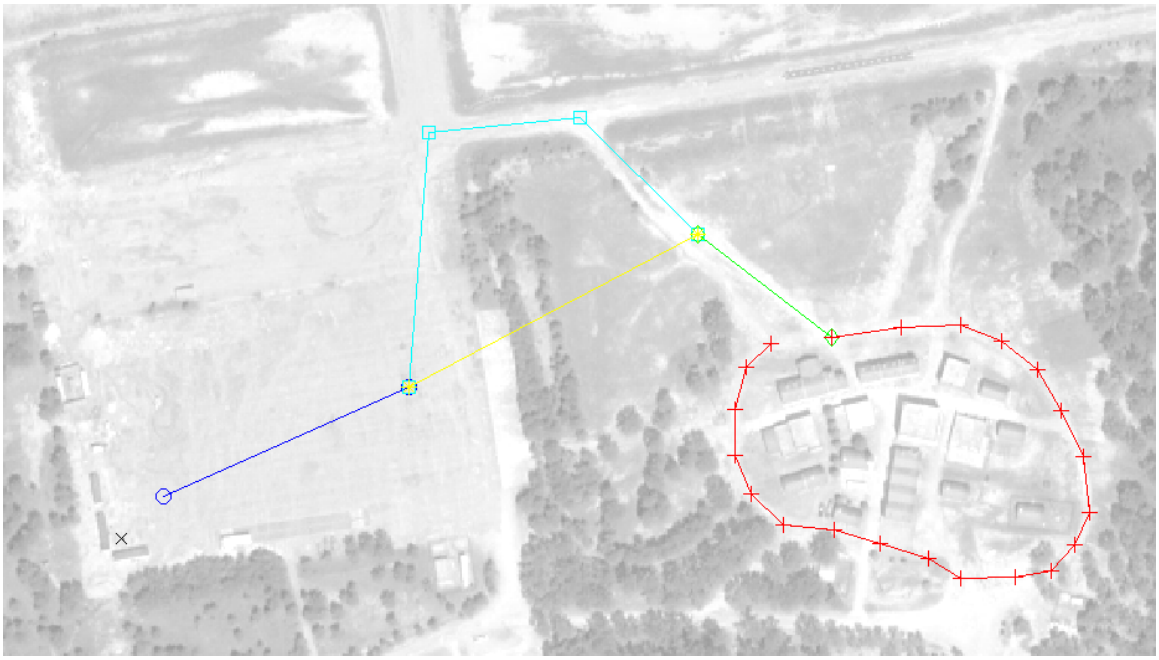


Figure 8: A non-intrusive surveillance mission. The network is tasked to provide surveillance information on an area of interest for some time.

These data rates were considerably chosen to be at these high levels. Utilizing appropriate compression algorithms and depending on the quality requirements, the necessary bandwidth for an acceptable video stream could be much lower, but higher rates were chosen in order to stress the network. It is not guaranteed at this point that any particular MANET can actually handle this network load.

4.2.2 General Motion Parameters in the Missions

All simulated missions make a couple of simplifying assumptions in order to keep the scenario generation less complex and to keep the focus on the MANET development, not on multi vehicle mission planning.

- All UAVs operate either at a fixed non-zero velocity or hover.
- All UAVs operate at the same altitude above ground, resulting in a 2D motion.
- No vehicle dynamics are modeled or simulated.

4.2.3 Scenario 1: Non-Intrusive Surveillance

This mission represents the most basic of the proposed reference missions. It intends to mimic a classic beyond line of sight (LoS) operation for either a civil or military environment. The task is to obtain surveillance information of an area of interest and relay that back to the control station. Fig. 8 pictures the flight plan for that mission. The control station is located in the lower left area of the picture, and the area of interest is given in the right area of the picture.

The mission utilizes eight UAVs in total. Two UAVs play a supportive role while the rest act in an active role. During the beginning of the mission, the UAV swarm approaches the area of interest in formation flight, represented by the blue and cyan colored path in Fig. 8. One UAV in the supportive role stays at the end of the blue path, one stays at the end of the cyan. From there the UAVs in the active role dispatch one after the other towards the area of interest, indicated by the green path. Once at the area of interest, the UAVs perform their continuous surveillance task. In this mission, this is given by flying a surveillance loop around the area of interest, indicated by the red path. After the active UAVs have gone through that loop several times, they return to the loiter position of the supportive UAV at the end of the cyan path and return to the control station from there (following the yellow path), again in formation flight. On their way back they pick up the second supportive UAV.

The loiter positions for the supportive UAVs are recreated as a static point in the simulation scenario, mimicking a UAV with hover capabilities, for example a helicopter. A small circle loiter pattern for a fixed wing UAV without hovering capabilities would also be possible but was not implemented in order to keep the mission generation less complex. The continuous surveillance loop was chosen over static holding points for helicopters for two reasons: in order to allow for fixed wing aircraft to perform the mission as well and to try to be more realistic. A continuous motion in this broader surveillance task provides a more complete overview due to a larger area under surveillance. This is comparable to a civil SAR mission in which, for example, an initial assessment has to be obtained after a chemical spill, earthquake, or wildfire, and the rescue teams need to compile this assessment

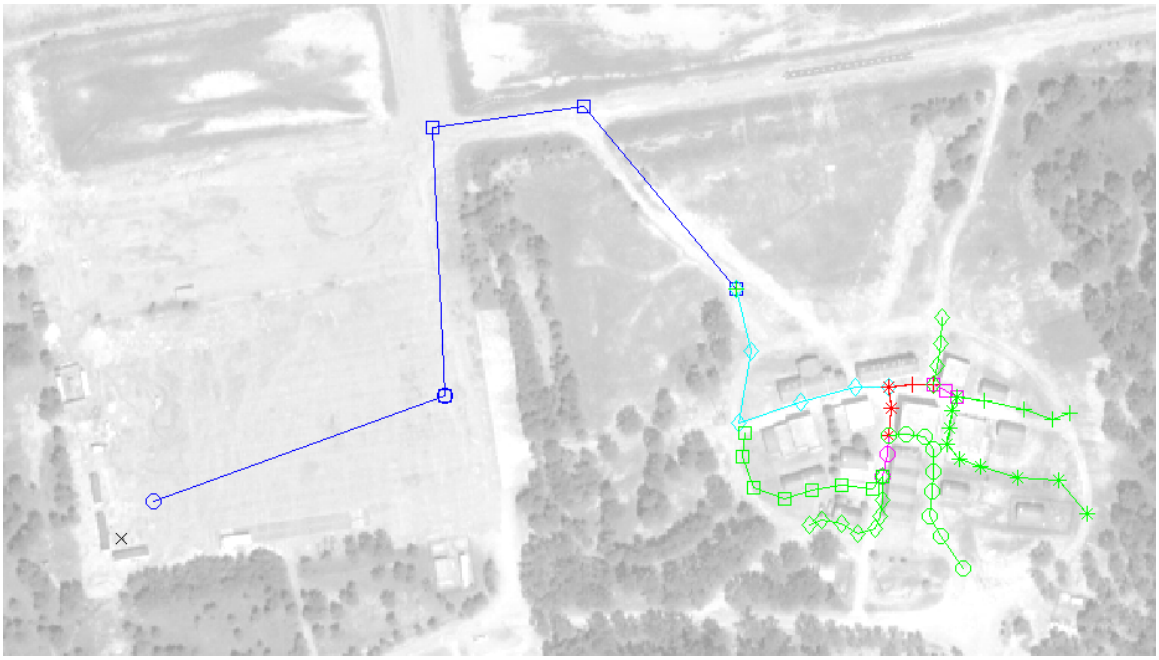


Figure 9: An expanding search mission. The network is tasked to perform an extensive search over an area of interest.

on a larger area.

The mission traffic situation is given by a single medium quality video stream during the formation flight, originating at the supportive UAV. Whenever an active UAV dispatches from the supportive UAV, this vehicle also transmits medium quality video. During the encircling of the area of interest, several still pictures are taken and transmitted back.

4.2.4 Scenario 2: Expanding Search

The expanding search mission pictured in Fig. 9 provides a natural extension of the non-intrusive surveillance mission. After an initial assessment has been obtained and further actions have been decided on, the need for a more detailed search arises. In this mission, the task is to map an area of interest in detail and return the gathered surveillance information.

The mission utilizes eight UAVs in total. Again, two UAVs play a supportive role, and the rest act in an active role. The start of the mission is identical to the non-intrusive mission at the control station in the lower left area of the picture. All eight UAVs take off from there and approach the area of interest in formation flight (leaving one supportive UAV at the first square marker). At about the same detach position as in the prior mission

(the end of the blue formation flight path), the active UAVs leave the supportive UAV behind and approach and enter the area of interest along the cyan path. Inside the area of interest, the active UAVs split up at the first intersection encountered (the red paths). By repeating this action the active UAVs branch out into the whole area of interest. Once single UAVs are isolated, they keep on going till they reach the boundaries of the area of interest and then return to the loitering supportive UAV at the detach point (green routes). Once all supportive UAVs are back at the detach point, the whole group returns to the control station in formation, picking up the second supportive UAV on the way.

Beside the loitering positions of the supportive UAVs, no other requirement for hover capability is present. However, as stated before, these hover points could also be circular loiter paths.

The mission traffic situation of this mission is the same as in mission one. During formation flight the supportive UAV transmit a medium bandwidth video. After dispatch, all active UAVs engage their medium quality data stream. During the sweep of the area of interest several still pictures are taken and downloaded to the control station.

4.2.5 Scenario 3: Coordinated Approach

The coordinated approach mission could be seen as yet another extension of the previous missions. After an initial overview has been gained (mission one) and the more detailed surveillance gathered in mission 2 has been post processed, a specific point of interest has been located and requires extensive surveillance. The task of the network is to reenter the area of interest, proceed to the point of interest and provide extended surveillance data to the control station. Fig. 10 showcases the flight trajectories for this mission.

The mission utilizes six UAVs, two in a supportive role and four in active ones, and tries to mimic a more military tailored setup. This is reflected by a more covert approach pattern and holding points at strategic locations. The group takes off at the control station, pictured in the lower left area of the picture, and approaches along the blue path towards the area of interest. The formation of the six vehicles is broken into two groups halfway to the area of interest, indicated by the cyan paths. Each group contains one supportive

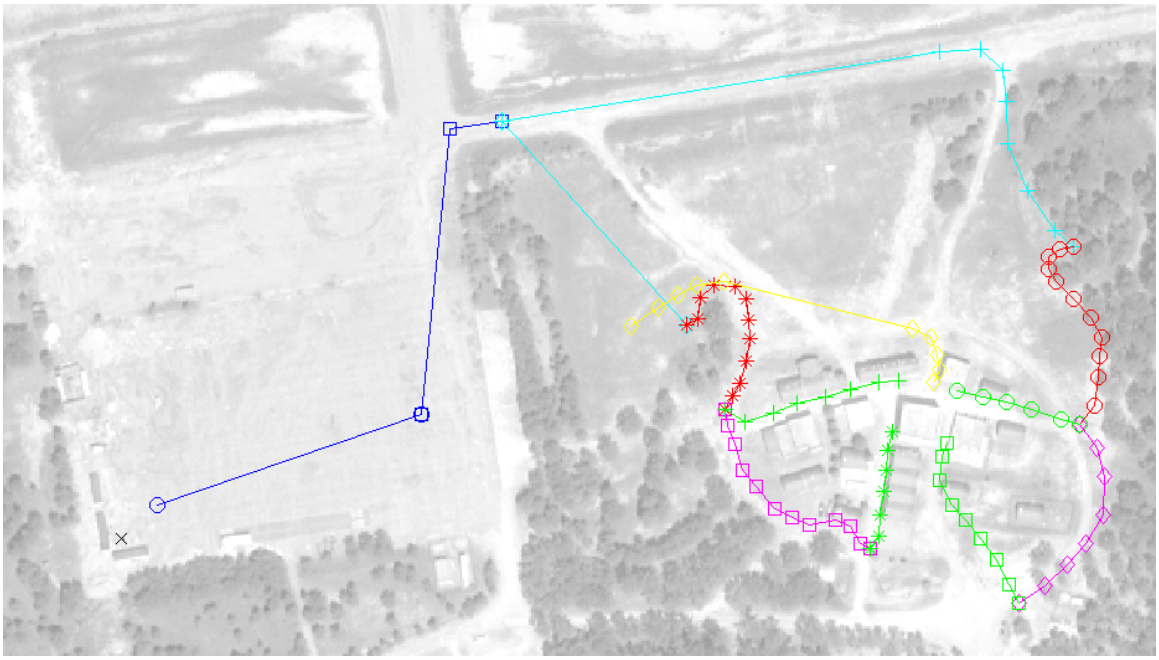


Figure 10: An coordinated approach mission.The network is tasked to approach a point of interest in a coordinated fashion and provide surveillance information.

UAV and two active UAVs and proceeds to the end of the cyan path, the loiter location for the supportive UAV of the respective group. From the loiter points, the active UAVs are dispatched along the red and magenta lines to the starting points of the green paths, taking an observing position at the perimeter of the area of interest. After a short pause, reflecting waiting on an external event, all four active UAVs engage towards the point of interest in the center of the area of interest. This part, along the green trajectories, is coordinated such that all four vehicles reach the point of interest at the same time. Once there the active UAVs perform a short hold, mimicking, for example, aerial support for a ground assault, and then evade the area of interest in formation along the yellow path. This motion is coordinated with the still loitering supportive UAVs such that all six vehicles reach a formation at the left end of the yellow line. The rest of the return to the control station is done in formation.

This scenario is tailored towards a helicopter swarm of UAVs. However, the loitering of the supportive UAVs could be a small circle, suitable for a fixed wing. Also, comparable to the non-intrusive surveillance mission, the hover at the point of interest could be altered

into a surveillance pattern, flying around the point of interest.

This mission has a slightly different mission traffic situation, tailored towards the high interest in a single point of interest. During the formation phases, only the supporting UAVs send out a medium quality video stream. When dispatched from the supportive UAVs, the active UAVs enable their medium quality video stream. While at the holding position at the perimeter of the area of interest, the supporting UAVs disable their video feeds and the active UAVs enable the high quality video feed. When the active UAVs are leaving the point of interest, the video feed quality is dropped back to medium. The supportive UAVs enable their medium quality feed and proceed to the formation forming point, at which the active UAVs as well as one of the supportive UAVs terminate their video feeds.

CHAPTER V

PROTOCOL DEVELOPMENT

The literature survey provided an initial overview on the different classes of MANETs currently developed. Given the nature of the system under consideration, a UAS MANET, the class of location-based routing protocols seemed to be the most appropriate class of MANET protocols for a development seed. This was not only stipulated by the dual use of the location information for both the MANET protocol and higher level processes such as collision avoidance, but also by the literature. [12] indicates that location based routing might be superior to reactive routing in VANETs.

5.1 Location Aided Routing

In the literature, two references popped up consistently: one to an initial paper on location aided routing (LAR), proposed in [19], and one to a MANET protocol labeled DREAM, proposed in [35].

The paper on LAR introduced the idea that the knowledge of location of a destination could be utilized to minimize the network topology to be searched in order to find an appropriate route. However, this assumes that the source knows the location of the destination. If this knowledge is not available, LAR reactively floods the network with a route request. This obviously creates a lot of overhead. To minimize overhead, a method in which location information could be piggybacked onto other traffic is mentioned, but not further specified. Beside this flooding for routes, LAR basically utilizes flooding into a geographically restricted area to reach the destination.

DREAM essentially utilizes a similar system of routing into a geographically constrained area. Besides using a different system to define this area, DREAM proactively floods the network continuously with location information. In order to avoid a network collapse due to the enormous amount of location packets, DREAM utilizes a dual approach: high frequency location information updates are only transmitted to close neighbors (in the physical

topology). The whole network is provided less frequently with location information updates.

Both approaches utilize flooding to distribute location information, a method not necessarily considered as optimal. In order to contain this flooding, [23] introduced a scalable location service, labeled grid location service (GLS). The intent is to allow for larger networks by cutting down on flooding. However, the proposed method utilizes an address-based location server system, and hence not all position information is known to all nodes.

5.1.1 Location Services

These three protocols essentially introduce the idea of a *location service* as part of the protocol logic to distribute location information throughout the network. LAR and DREAM both distribute this information globally: LAR as a reactive pull message, and DREAM as a proactive push message. GLS keeps that information centrally stored and utilizes a reactive approach as well.

Since the initial idea to utilize a location-based protocol also was based on the dual use as collision avoidance data, the reactive pull approaches are suboptimal for both the global as well as the compartmentalized approach proposed with GLS. In order to allow for an autonomous collision avoidance system onboard the UAVs, the vehicles must know the position of their direct neighbors as well as the position of the vehicles along their flight path. In the compartmentalized approach of GLS, this information would be accessible through an appropriate location server - as a pull information. Hence, a constantly moving UAV would have to query the location server for more or less all vehicles' positions. However, given the sparse nature of UAS networks, this seemed to create several issues, such as bandwidth bottlenecks, and as such this approach, though it promised a higher scalability, was dropped. In addition, the beneficial feature of scalability seemed to be rendered useless since the location information would have to be broadcast all the time. In this method, location information is only pulled by the UAVs and delivered through the location servers instead of everybody. This seemed to be an inferior approach to a proactive flooding-based one. However, a simulation or thorough investigation has not been performed.

5.1.2 Performance of Location Aided Protocols

Since GLS and LAR were dropped due to their reactive nature, a performance evaluation on location-aided MANET protocols utilizing a proactive location service was necessary. During the initial literature survey, several sources dealing with MANET performance were identified, and a focused investigation was conducted.

Dr. Jeff Boleng investigated several aspects of MANET performance in his dissertation ([8]). As Boleng stated there, "The goal is not to create Yet Another MANET Protocol (YAMP). Instead, the goal is to optimize existing protocols and to create a mechanism to enable the combination of multiple protocols into a hybrid protocol.". This matches the approach of this work. In order to establish a baseline, the performance of LAR and DREAM were investigated.

[10] gives a direct performance comparison of these two protocols. Unfortunately, one of the conclusions found therein states that the complexity of DREAM does not appear to provide benefits over a simple flood. This results from the frequent use of a global flooding as a recovery procedure. DREAM utilizes this approach if a regular route discovery could not be completed successfully. However, [10] also states that location information enhances the DSR protocol and gives hope to pursue location aided routing.

Further research revealed [31], a work on improvements on DREAM. The proposed improved DREAM (iDREAM) includes an estimated future position into the broadcasted location information. This information then is utilized to better estimate the actual position of a node (based on an interpolated estimated position calculated from this new information) in order to optimize the route finding procedure. The intent was to minimize the fall back to the recovery flooding routine of DREAM and hence increase the performance over a simple flood. Unfortunately, the results only showed a marginal performance increase in very specific average node speed situations. Hence, iDREAM cannot be considered to generally perform better than DREAM.

5.2 Initial Tests

For initial testing, a diverse approach was taken. In order to (re-) evaluate the literature findings, the evaluation scenarios were implemented in a simulation environment. This idea was stipulated by the initial assumptions on how a UAS scenario would affect the performance of a MANET, outlined in Sec. 4.1, and intended to verify the stated findings for UAS environments.

5.2.1 The Network Simulator `ns2`

In order to obtain comparable results to the literature, the network simulator `ns2` was chosen as a network simulation and development environment. According to [22], `ns2` is the tool of choice in a majority of MANET-related, literature and should allow for results checkable against literature findings.

The open source software `ns2`, available from [13], provides an environment consisting of a compiled C++ simulator core that is controlled during a simulation by oTcl scripts. oTcl is an object-enabled expansion of the script language Tcl. It is available via [41].

The utilization of a proven simulator should ensure representative findings in order to better judge the performance of different MANET protocols, since it provides simulation capabilities related to networking areas outside the scope of this work.

Furthermore, `ns2` provides the ad-hoc protocols DSR, DSDV, TORA, and AODV as part of the `ns-allinone` package. This allows for a quick comparison with these baseline ad-hoc protocols, both reactive and proactive.

5.2.2 The Visualizer iNSpect

Since `ns2` is a shell based tool without a graphical user interface, a visualizer was necessary to validate the conversion of the reference scenarios into `ns2` compatible scripts. Besides the `ns2` accompanying tools, *iNSpect* ([38]) was chosen to visualize the network. This tool of the Toiler's Research Group provides motion animation, connectivity graphs, and a graphical overlay of network traffic onto the motion animation.

5.2.3 First Lessons Learned

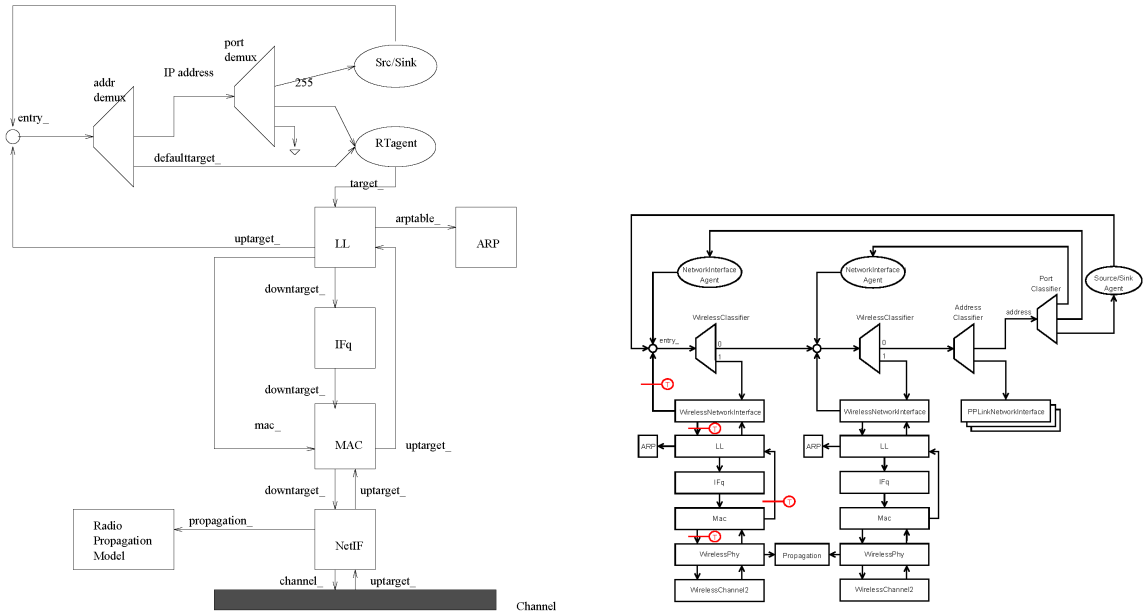
After the reference scenarios had been ported to `ns2`, a first simulation was performed utilizing the DSR protocol and a much smaller network traffic load in order to keep the simulation output small and comprehensible. Only one medium bandwidth video stream originating at an active UAV was simulated. DSR was chosen since [30] indicates a performance benefit of DSR over AODV in scenarios comparable to the reference scenarios. In the simulation, the UAVs utilized WiFi equipment according to the IEEE 802.11b standard. Furthermore, Tx power settings were matched with those utilized by the UAVRF.

Two main lessons were learned from this initial simulation. Since the UAVs utilized the permissable Tx power settings, nearly all UAVs were in range of each other during the execution of the mission. (A two-ray ground propagation model was utilized to simulate the radio coverage area.) As a result, very few route changes occurred (compared to the expected amount). Besides the finding that the need for a MANET protocol could be eliminated by utilizing a star topology and high(er) power WiFi systems ([11]), this stipulated the idea of looking more into proactive protocols, as it seemed that the traffic generated in order to update routing tables would be limited.

Validating the simulation against the evaluation metrics lead to a second result: all elements in the metric seemed to be feasible by adopting the MANET protocol, leading to a newly proposed protocol. Unfortunately, the feasibility of several different network interfaces aboard one UAV seems doubtful. Both the MANET protocols and `ns2` appeared to be optimized for a single interface per node only.

5.2.4 The First Development Iteration

The requirements related to a future implementation explicitly state the compatibility to environments with several different interfaces. This is especially true for scenarios in which some vehicles utilize incompatible WiFi equipment - for example wireless serial and WLAN- and, therefore, need to make use of vehicles equipped with communication interfaces for both systems. This is a realistic scenario for the UAVRF.



(a) The native MobileNode structure in `ns2`.

(b) The extended `mw-node` structure with two interfaces.

Figure 11: `ns2` does not natively support multiple wireless network interfaces connected to one node. The `mw-node` patch redefines the wireless node representation in a modular way, allowing for a much more versatile framework within `ns2`.

5.2.4.1 Adaptations of `ns2`

The network simulator `ns2` utilizes a structure labeled "mobile node" (Fig. 11(a)) for simulations involving wireless networks. This structure, developed by the CMU Monarch Project, only allows for one wireless interface per mobile node. In order to overcome this, the module-based wireless node patch (`mw-node`, [26]) was applied to `ns2`.

The modular structure of the `mw-node`, pictured in Fig. 11(b), allows for per-interface routing routines as well as per-node routing routines. On top of that, all related modules can be changed and adapted. This allows for different interface cards to transmit with different physical characteristics, like Tx power, frequency, and sensing and receiving thresholds. Furthermore, if desired, different RF propagation methods could be utilized for the different network interfaces.

5.2.4.2 Adaptations of the MANET Protocol

The challenge of different network interfaces creates a whole new angle on MANET systems and protocols. If one node has several interfaces, several options are possible:

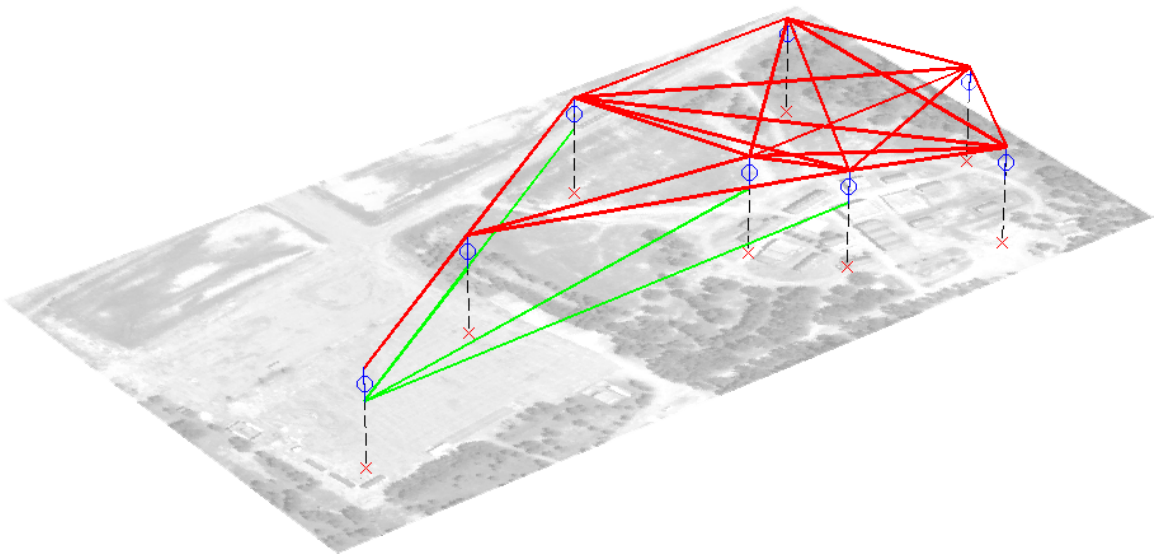


Figure 12: Due to the inaccessibility of different network interfaces to the RF signals of other WiFi systems the network is partitioned. These partitions manifest as layers on top of each other. Nodes with the capability to access several layers act as gateways. In this picture, two layers are present.

Picking One Interface for the transmission at hand would be one option. This raises the questions of how the routing logic should select an interface and how the different interfaces actually affect the network topology.

Transmitting On Both Interfaces is another option. Though this is a simple and easy to implement approach, the benefits seem to be doubtful. This approach overcomes issues due to a network partition because of the different interface systems, but it also creates a lot of congestion simply due to the fact that packets are transmitted in a WiFi spectrum which might not even be accessible by the recipient.

One Routing Logic Per Interface would not require any adaptation of currently existing single interface MANET protocols. However, some logic needs to be created to interconnect the parts of the network which are separated due to different WiFi technology.

5.3 Network Partitioning Due to Layering

The presence of several different network interfaces with incompatible RF specifications leads to a segmentation of the network. Each WiFi system creates a separate layer within

the network topology. Nodes utilizing network interfaces operating in that particular WiFi system are on that layer in the topology. In Fig. 12, this situation is pictured for two layers. Nodes operating a WLAN interface are able to connect to other nodes operating WLAN, indicated by the red lines. Some nodes also utilize wireless serial connections at the same time and can connect to other nodes on that layer, indicated by the green lines. The nodes carrying network interfaces for both layers could reroute traffic onto the other layer. This leaves nodes with a much more complex routing situation since now several options across several layers might be physically feasible. A new metric for this has to be created.

5.3.1 Cross-Layer Routing Metrics

Previously, several metrics have been introduced in Sec. 2.3: shortest path, throughput optimization, energy awareness, and security awareness. These routing metrics have to be expanded in order to compare possible routes which could go across layers. The metrics are still valid. The challenge, though, is to redefine the metrics on how to compare possible connections on the different layers. However, if the layered network topology, as pictured in Fig. 12, for example, is looked at globally, all that is necessary to maintain the already established metrics is to assign an appropriate cost to each possible connection, independently of the layer this connection is on. Doing this allows an objective comparison of all physically possible connections with respect to any of the introduced metrics.

5.4 A Global Approach to Routing

Based on the network layer partitioning, a global approach for a MANET protocol is taken. As mentioned before, if the routing logic has a global knowledge of the complete network topology, including all associated costs, then a globally optimal route can be found. However, a global approach not only allows for several layers, but also avoids problems of local minima in the route finding process. The problem of local minima has been formulated in several places and comes naturally with local optimization routines. Though the problem is solved in MANETs, some assumptions in specialized protocols might not be given in a UAS environment. Greedy perimeter algorithms, for example [18], are able to avoid local minima, but require a dense network topology - a condition not necessarily given in a UAS

MANET.

A global approach provides several substantial benefits for UAS MANETs. Global knowledge of the network topology is beneficial for (multi-layer) routing. Global knowledge of the physical topology is beneficial for the operation of a UAS, for example for collision avoidance algorithms based on sense-and-avoid. Third, global knowledge on the status of the UAS is necessary at least at the control station in order to provide situational awareness to the human operator.

5.4.1 Suitability of a Global Approach

Certain constraints arise with a global approach, though. Due to the nature of global knowledge at every node, a global system does not scale to a larger network. Furthermore, global data distribution provides for a large overhead, if the mission data (the payload of the network traffic) is compared to the total traffic generated. However, these constraints are not necessarily as severe in a UAS environment.

Current UAS are not networks of hundreds of vehicles. As a matter of fact, constraining a UAS to an upper limit of 20 vehicles seems to be a realistic scenario, at least compared to the real applications of UAS today. Twenty nodes, though, do not provide for concerns about severe computational problems or bandwidth bottlenecks within a network, given the usable bandwidth of IEEE 802.11g WiFi systems (stated in [43] to be 19Mbit/s for a network utilized by several nodes) and the messages previously stated in Sec. 4.2.1. Based on these messages, a generated traffic of about 7kbit/s per node is estimated without the sensor data streams of 300kbit/s or 1Mbit/s. Even if all nodes are assumed to be (topologically) lined up, creating a chained network topology, this would theoretically allow for all nodes to send the medium bandwidth data stream and their update messages and even for nearly all nodes to send high bandwidth data streams.

The issue of overhead also does not really apply to a UAS MANET, given certain provisions. Overhead is defined as network traffic necessary to perform regulatory or controlling actions. In a MANET, route request procedures as well as route maintenance procedures would create overhead. Mission and control traffic only partially add overhead in form of the

appropriate packet headers, possible checksums at the end of a packet, or acknowledgment packets being sent. If data necessary for controlling the UAS could be utilized for routing procedures, the calculated overhead would dramatically shrink. This is mainly mathematically, though, since packets which earlier counted towards overhead now are counted as control traffic.

5.5 The DBGR Framework

Based on the previous findings, requirements, and reference scenarios, a new flavor of a LAR protocol is proposed. The framework, labeled DBGR for DREAM Based Graph Routing, introduces a globally optimized routing strategy based on two independently operating components and a local database. A *location service* provides means for every node to obtain location information of all other nodes. A *routing agent* calculates globally optimized routes using this data as well as a local database.

5.5.1 The DREAM Based Location Service

The notion DREAM location service (DLS) appears in an implementation of the DREAM protocol for `ns2`, accompanying the related paper [10]. There, DLS refers to the mechanism utilized in the DREAM protocol implementation ([37]) to distribute location information.

In the proposed DBGR framework, DLS describes any location service based on the idea of a distance effect. This effect, introduced to LAR protocols by DREAM, describes the subjective difference in relative velocity of two objects, based on the distance between them. Since relative velocity could be described as an angular velocity of one object relative to another object, closer objects appear to be much faster than objects further away. This observation translates into the context of MANETs. Position updates of objects closer to an observing node have to occur at a higher frequency than updates for objects further away. This is true if the location information is used for routing and it is true if the location information is used for application level processes, such as collision avoidance.

In the DBGR framework, DLS describes a high level location service. Following the basic idea in the implementation of the DREAM protocol, DLS provides nodes closer to the ownship more often with position updates than nodes further away from the ownship.

The specifics of how this is implemented and what governs the update frequency have to be determined based on the implementation.

5.5.2 The Local Database

DLS provides location information and keeps this data current. In conjunction with that, a local database (LDB) on each node contains time invariant information about all other nodes currently in the network. This information would be proactively broadcast by any node joining the network and hence constitute an extended `hello` message which is used to instantiate this node. This would fulfill the instantiation requirement in Sec. 3.1.1.

The LDB could contain mission specific data, but it has to contain certain information necessary for DBGR to function. An essential data set contains the number and kind of network interfaces carried as well as the associated power settings and antenna gains. Furthermore, an indicator for the complexity of the operation to reroute data from one interface the another is needed.

5.5.3 Using Data: The Routing Agent

The second active part of the DBGR framework is the routing agent. The agent assumes that DLS provides a table containing location information of all nodes in the network. Based on this physical topology, the routing agent estimates the current network topology and computes an optimized route to the destination.

The routing agent is able to determine the number of layers of the network topology from the number of different WiFi systems in the LDB. For each layer, the routing agent computes a connectivity graph. This is done by estimating physically possible connections based on the location information from DLS and a connectivity calculation based on an internally stored RF propagation approximation and the Rx/Tx data of the involved notes, taken from the LDB.

Based on an applicable metric, the routing agent determines the cost associated with each assumed physically possible connection. If a node can act as a gateway between layers, the appropriate connections are created in the connectivity graph and a cost based on the complexity entry in the LDB is associated.

The last step is to calculate the optimal route from the source to the destination through the connectivity graph. This could be done by an appropriate graph theory algorithm, for example Dijkstra's shortest path algorithm.

CHAPTER VI

IMPLEMENTATION DRAFT FOR DBGR

The DBGR framework outlined in Sec. 5.5 is given in very general terms. This chapter proposes a first draft for a future implementation.

6.1 Initial Configuration

In order to prepare each node for participation in the network, DBGR relevant parameters need to be configured at each node. This information should be sufficient in order for DBGR to work and to maintain a network without human intervention.

Listing 6.1: Network Interface Related Data

```
class iface
{
    iface_ID;          // unique interface ID
    iface_Name;        // descriptive interface name
    frequency;         // frequency of operation, in MHz
    layer;             // network layer this interface can access
    tx_Power_max;      // Allowable Tx power in mW
    tx_variable;       // TRUE for interfaces with variable Tx power
    tx_Power_NOW;      // actual Tx power setting for variable
       systems
    rx_Threshold;      // Rx receiving threshold in mW
    antenna_gain;      // Rx/Tx gain in dB
    complexity;        // cost of transferring data to/from the
       application layer
}
```

This set of node-specific data will be broadcast during the initialization. A set of basic parameters is proposed in Lst. 6.1.

6.2 Instantiation

When a node joins the network, an instantiation is performed. During this process, the node broadcasts its interface information (Lst. 6.1) and other specifics as outlined in Sec. 3.1.1. This process is considered a handshake between the node and the network and should ensure

that the network and especially the control station operator are aware of the presence of the new node.

6.3 Regular Operation

During regular operation of a node within the network, several tasks have to happen in order for DBGR to work. The DLS has to provide ownship location information to the network as well as process incoming location information in order to provide the location table to the routing agent. The routing agent then has to process this location table, find a route to the destination in case a packet needs to be sent, and process incoming data packets.

All these tasks happen in the application layer. As a generic interface, we declare two functions that act as an gateway layer between the network interface and the application.

Listing 6.2: Gateway Methods to and from the Network Interface

```
void iface_receive(packet)
{
  if is_dls_packet(packet)
  { // the packet is associated with DLS
    dls_receive(packet);
    return;
  }
  else
  { // the packet is a control or mission packet
    routingagent_receive(packet);
    return;
  }
}

void iface_send(packet)
{ // pass the packet to the appropriate network interface
  if (packet.iface_id == BROADCAST)
  { // broadcast packets are send on all interfaces
    for (id = 0; id in node.iface_list; i++)
    {
      iface(id).transmitt(packet, now+jitter);
    }
  }
  else
  { // the packet is send on a certain network interface
    iface(packet.iface_id).transmitt(packet, now);
  }
}
```

6.3.1 The DLS Methods

DLS has to provide two main methods, dls_send(packet); and dls_receive (packet);. Since protocol traffic is dually used also for collision avoidance, in this draft the messages DLS creates are identical to the messages introduced in Sec. 4.2.1. This matches an initial approach to the use of the distance effect, stating that closer nodes need updates more often than nodes further away. Hence, this draft declares two zones around each node. Nodes in the *near zone* are provided with the short and fast protocol message. Nodes in the *far zone* (and obviously to the ones in the *near zone* also) are provided with the longer and less frequent protocol message. On top of that, both messages are sent to the control station.

The *near zone* is initially declared as a sphere with radius `near_zone_radius`. How this radius is set could be based on the current velocity, for example.

Listing 6.3: DLS Send Method

```
1  void dls_send( )
   {
       dls_pkt = creat_new_packet();

5      // update information to be send out
       node.update_states;

       // create a short or long message
       if (send_short_packet)
10     { // short and more frequent packet
         dls_pkt.type = SHORT;
         dls_pkt.near_zone_radius = node.get_nzr();
         dls_pkt.control_data = node.control(SHORT);
       }
15     else
       { //long and less frequent packet
         dls_pkt.type = LONG;
         dls_pkt.near_zone_radius = INF; // flood the network
         dls_pkt.control_data = node.control(LONG);
20     }

       // set information to be put into the DLS packet
       dls_pkt.source = OWN_NODE_ID;
       dls_pkt.destination = GCS;
25     dls_pkt.pos_current = node.position(now);
       dls_pkt.pos_future = node.position(now+delta_t);
       dls_pkt.route = routing_agent.route(GCS);
```

```
28      dls_pkt.iface_id = BROADCAST;

30      // send the packet
        iface_send(dls_pkt);

        return;
    }
```

The counterpart to this is the dls_receive(packet); method. As soon as a received packet is determined to be a DLS packet, this method gets called. The call to routing_agent .route(); in Line 27 returns a route to the control station (or any other destination) from the ownship. This method is provided by the routing agent.

Listing 6.4: DLS Receive Method

```
1  void dls_receive(packet)
   {
     // check if the packet has been received earlier
     if (received_packets.check(packet)) return;
5
     // since the information has been received, it might as well be
         used
     update_location_table(packet);
     received_packets.add(packet); // avoid future duplicates

10   // determine if and how this packet has to be forwarded
     if (packet.type==SHORT)
     {
       if (routing_agent.distance(packet.source) < packet.
           near_zone_radius || OWN_NODE_ID in packet.route_to_gcs)
14     {
15       packet.iface_id = BROADCAST;
         iface_send(packet);
         routing_agent.route_check(packet);
18       return;
       }
20   }
     if (packet.type==LONG)
     {
       packet.iface_id = BROADCAST;
       iface_send(packet);
25     routing_agent.route_check(packet);
26     return;
     }
   }
```

In Line 13, the forwarding logic for `SHORT` DLS packets is given. These packets are either

Table 1: The connection matrix represents a connected graph of the network. Each node (1,2) has two layers (WLAN, Serial) and the matrix defines the cost between them. No connection is represented by ∞, 0 is no cost (on the main diagonal), *Comp.* represents the complexity to go from a layer to the node (the application layer), and *Cost* the cost of an assumed-to-be-possible connection between two nodes on the same layer. The matrix is not necessarily symmetric.

	1	1W	1S	2	2W	2S
1	0	Comp.	Comp.	∞	∞	∞
1W	Comp.	0	∞	∞	Cost	∞
1S	Comp.	∞	0	∞	∞	Cost
2	∞	∞	∞	0	Comp.	Comp.
2W	∞	Cost	∞	Comp.	0	∞
2S	∞	∞	Cost	Comp.	∞	0

forwarded if the receiving node is inside the near zone or a part of the route to the control station. `LONG` DLS packets are always forwarded.

6.3.2 Routing Agent Methods

The other active part in DBGR is the routing agent. This part of the code utilizes the location information provided by DLS as well as a local database, which contains the information gathered during the instantiation of the other nodes.

6.3.2.1 Estimating the Network Topology

Based on the physical topology and the information about the WiFi equipment of the different nodes, the routing agent creates a graph, representing the estimated network topology. This graph is realized by means of a *reachability matrix*. This matrix is then updated with the costs for each estimated connection, creating the *connection matrix*.

The connection matrix (Tbl. 1), is then used to find a globally optimized route. It is optimized and not optimal since the connection matrix does not reflect the current network topology, only the estimated one.

6.3.2.2 Calculating an Optimized Route

If the correct cost has been applied during the creation of the connectivity matrix, finding a globally optimized route is reduced to a shortest path or traveling-salesman-like problem. An initial implementation could be based on the Dijkstra algorithm. In App. A a proof of

concept is given in MATLAB code.

The MATLAB code of the routing agent is a straightforward implementation of the ideas presented so far and uses the MATLAB implementation of Dijkstra's algorithm. The provided algorithm either calculates the shortest routes from a given source to a given destination or it gives the shortest routes from the source to all other nodes in the graph. The latter executes faster than the former and total execution time appears to be acceptably short.

6.3.2.3 Estimating the Connection Cost

In order to do all this, the conversion from the reachability matrix to the connection matrix requires the calculation of an estimated cost for each assume-to-be-possible connection. These costs directly relate to the routing metrics stated in Sec. 2.3.

The shortest path metric is straight forward. Each connection in the reachability matrix is assigned a fixed cost, making all connections equally costly, and a cost minimization ends up with the shortest path (with respect to the number of hops).

The energy-aware metric is also comparably simple. If the local database indicates a network interface to provide variable Tx power, the routing algorithm can compute that power level by simulating the RF propagation and obtain the necessary Tx power in order to match the Rx threshold at the destination. This power is then assigned as cost to this connection. Since the exact distances and locations are not known, though, a fudge factor has to be applied in order to ensure a delivery to the destination[1]. If a network interface does not provide variable Tx power the default setting from the local database is assigned. The security aware metric could utilize a similar approach.

These metrics have in common that they are independent of any other connection a node along the route maintains. The throughput optimizing metric does not allow for this and, theoretically, the bandwidth utilization at each would have to be known in order to compute an optimized route. In the draft implementation described so far, this would only

[1] A more complex scheme could incorporate the fact that every network interface has a sensing threshold which is lower than the receiving threshold and hence could detect that something is send, though it cannot read it correctly.

be possible if the bandwidth utilization would be announced to the network. This could be done, for example, by putting a related marker in the DLS packages.

However, these packets might not be transmitted frequently enough, and another estimation scheme has to be applied. For this draft, an estimate based on the number of possible connections per node is proposed.

Listing 6.5: Estimating Available Bandwidth

```
cost estimate_bandwidth(source,destination,layer)
{
  // count all assumed to be possible connection at a node
  int src_conn = get_connections_at_node(source,
     reachabilit_matrix);
  int dst_conn = get_connections_at_node(destination,
     reachabilit_matrix);

  // estimate the available bandwidth for this connection
  bandwidth = 2*bandwidth_of_layer(layer)/(src_conn+dst_conn); //
      in Mbit/s

  // transform bandwidth into a normalized cost and return
  return cost=get_normalized_cost(bandwidth);
}

cost get_normalized_cost(bandwidth)
{ // convert the bandwidth using an exponential scale

  // X-Axis = bandwidth
  // Y-Axis = cost
  // find the function that goes through two points
  // point1 (MIN_BANDWIDTH, MAX_COST)
  // point2 (MAX_BANDWIDTH, MIN_COST)
  // Y = e^-[a(X-b)]

  log_temp = log(MAX_COST)/log(MIN_COST);
  b = (log_temp*MAX_BANDWIDTH - MIN_BANDWIDTH)/(log_temp-1);
  a = log(MAX_COST)/(_temp-MIN_BANDWIDTH);

  return cost = exp(-a*(bandwidth-b));
}
```

The optimization scheme given in Lst. 6.5 is by no means optimal, but outlines a possible approach for estimating bandwidth utilization based on the network topology only. The approach assumes an equally utilized network, i.e. the same amount of traffic is present on each possible connection. The algorithm then approximates the available bandwidth of a

node based on the utilization of the medium in the area close to the node.

If a connection is (assumed to be) possible between two nodes `A` and `B`, these two nodes can overhear all outgoing traffic of the other node, respectively. This is independent of whether or not they are the actual addressee of the overheard traffic. While node `A` overhears outgoing traffic on the one-hop neighbor `B`, the wireless medium in the area around `A` is partially blocked for usage by `A` since any transmission attempt would most probably result in an interference-based packet drop. The algorithm presented in Lst. 6.5 hence does not actually optimize the throughput, but finds the route with the (estimated) highest probability for an unused wireless medium.

As stated before, this is by no means optimal. If nodes `A` and `C` are close to each other and both want to transmit to the far away located node `B`, the routing agents on `A` and `C` would most probably create almost identical routes. If no other network traffic is assumed, this would lead to a solution close to the worst case solution of an identical route. However, if only one node utilizes this metric, for example the sole node transmitting a high bandwidth data stream, a bandwidth optimized routing solution can be achieved.

6.3.3 Route Adaptation

In Line 17 and Line 25 of Lst. 6.4 a call is made to the routing agent method routing_agent .route_check(); to check the route in the packet header. If the node that forwards a packet assumes to have a more beneficial route to the destination than the one specified in the packet header, the node updates the route in the packet header and notifies the original source of that packet about this change.

This allows for an simple route adaptation process. If a packet is sent to a destination far away from the source, the location information available to the routing agent can, and most likely will be, outdated. Even though the routing agent tries to compensate for that by utilizing the information on a future position of a node (sent out in the DLS packets), locations could be off. Given the idea of the distance effect, nodes closer to the destination than the original source will have more recent location information. Hence, while the packet travels towards its destination, the forwarding nodes have more and more accurate

information about the destination's position and can hence calculate a better route.

Obviously in the current implementation with only two zones, the near zone and the far zone, this system is lacking and surely not the most efficient route adaptation process. However, this adaptation allows for future optimization. Either by introducing more zones or by replacing this system with a completely different approach.

CHAPTER VII

CONCLUSIONS AND REMARKS

This work proposes DBGR, a new framework for MANETs in UAS environments. DBGR provides an optimized architecture comprised of three components: a location service, a routing agent, and a node local database. DBGR describes how these parts connect and interact and outlines their basic functionality, therefore allowing for an independent development and improvement of each component. As an initial core for an implementation effort, this work proposes to mimic the location service of DREAM and to utilize Dijkstra's algorithm for route computations in the routing agent.

7.1 Findings

One of the main findings of this work is the necessity for a dual-use of network packets and, as outlined in Sec. 5.4.1, the notion of network overhead is closely related to this. Utilizing packets and their content information not only in the maintenance of the DBGR framework and actual data transmission, but also as valuable information for the GNC processes, reduces the overhead substantially. The identification of location information as a common denominator for both MANET routing as well as UAS GNC allowed for reduced network traffic and also less network overhead.

Following a similar argument, a global approach to routing has been proposed for DBGR, a concept not generally assumed to be beneficial in MANETs. However, due to the nature of UAS and the following requirements and constrains, global knowledge appears to be beneficial, if not necessary. As outlined in Sec. 5.4, a global approach is advantageous for a "globally optimized" routing process, and, on the same note, global knowledge of the UAS status can be of assistance for certain GNC processes, such as collision avoidance in a sense-and-avoid situation.

However, a global concept does not scale and larger networks might need to alter this concept. The proposed reference scenarios do not indicate the need to limit the horizon of

a node's knowledge on the physical topology to somewhere below global, as current UAS do not utilize the large number of actors requiring higher scales.

Generally speaking the interconnection of GNC and MANET processes seems to be beneficial to both entities. The MANET protocol could benefit from the knowledge of future motion throughout the network. This is a rather uncritical interaction as the operational safety is not challenged at any time. The reverse interaction, the MANET affecting GNC processes, however, poses much more serious threats regarding operational safety, but also promises much higher benefits. As mentioned in Sec. 3.3, if the MANET protocol could cause GNC actions such as physical relocation of a node, thus reducing the workload on an operator by cutting back on actions related to maintaining the network. This, however, enters the field of general human-machine-interface challenges and poses a detached research field - extending the machine side of that with the network as an actor.

7.2 Open Challenges

The literature review leading to the development of DBGR unveiled several open challenges, some rooted in basic MANET or networking principals, some arising from MANET simulation efforts, and some originating from the UAS setting.

The first two groups are most often related. Examples are realistic propagation models for radio waves in an urban environment or open landscapes or challenges related to the MAC layer simulation of IEEE 802.11. In the latter, the same challenge, the shared medium access problem, coexists as a simulation issue as well as a routing issue. The routing issue is related to performance increases due to minimized packet collisions, i.e. the routing agent needs to find a route with minimized packet collisions. This example indicates the close relation between the accuracy of the simulation and the capabilities of a MANET protocol and underlines the need for future work in increasing the capabilities and accuracy of network simulators.

Another challenge is the deteriorated performance of TCP in MANETs. Unaltered TCP does not seem to be an appropriate choice in a multihop environment, even with the RTS/CTS handshake ([44]) disabled. UDP steps in as an immediate replacement, but this

results in challenges related to confirming the delivery of packets to the destination node. As it appears, using UPD, this feature has to be moved from the lower layers of the OSI model to the application level. This upwards movement of networking processes to the application level also seems to be necessary in order to allow for several inherently incompatible WiFi systems to harmonically coexist within a single UAS. Achieving this was stated as one of the requirements and led to the introduction of the layered approach in routing.

On the list of challenges related to the UAS setting, the previously mentioned global knowledge of the physical network topology has to be mentioned. Though the proposed scenarios do not indicate the need for a receding horizon approach, combining this with the DLS update ranges might be an interesting route to investigate.

Also related to this is the challenge to develop an appropriate human-machine-interface for UAS. Reducing the workload on an operator controlling several UAVs at once, either as a swarm, individually, or both, seems to be essential in order to field UAS consisting of several UAVs.

Another topic that needs to be looked into is the issue of addressing network nodes. The DBGR framework allows for additions of new network nodes during network runtime. However, DBGR assumes some initial configuration has been done at each node prior to adding it to the network and the issue of node addresses, though briefly mentioned in Sec. 3.1, has been skipped. In a small network this manual assignment of node addresses might be possible, but larger networks call for a DHCP-like system. A smart addressing system could also be useful in the general scalability challenge of the DBGR framework.

Outside the framework-related work, other topics still remain untouched by this work. On the simulation level of the UAS, both off-line as well as internally, the compliance to HLA has to be investigated for a future implementation of DBGR, for example, in GUST. HLA promises to be a beneficial framework for communication with and within a simulation environment and hence should be used as an implementation baseline. This should enable the participation of UAVRF UAVs in large scale simulations and also allow for a smooth integration of simulated entities into a HITL test.

JAUS states details on data message structures, like packet headers and formats of the

transmitted data. Hence, JAUS is a completely detached requirement and not necessarily relevant for an implementation effort. A JAUS compliant implementation can be obtained independently of the network protocol inside a UAS if an appropriate gateway node is able to translate UAS internal traffic into a JAUS format.

This points out another topic for future work: the connection of a UAS MANET to other (wired) networks. Since the UAS network most probably utilizes a non-standard networking scheme, a gateway node is necessary. This node has to deal with the challenges related to keeping the (wired) network outside the UAS and the (wireless) UAS MANET in synchronization.

7.3 Other Contributions

This work also proposes three reference scenarios for MANET evaluation in a UAS environment. As outlined in Chp. 4, UAS require different evaluation scenarios in order to mimic realistic UAS behavior. The scenarios presented in this work represent realistically scaled UAS missions, tailored to the current main utilization of UAS in the research community and the armed forces.

The scenarios presented here cover UAS specific aspects of the nodes involved and their networking capabilities (Sec. 4.1), the motion behavior during a realistic mission (Sec. 4.1.2), and the network traffic during these missions (Sec. 4.1.3).

The scenarios can be utilized in different network simulators (like `ns2` or QualNet) and, hence, allow for a performance comparison of different MANET protocols in a UAS environment.

APPENDIX A

MATLAB ROUTING AGENT DRAFT

This section presents the MATLAB code of a draft implementation of the DBGR routing agent. The function plots a background map (BKG) and the user places nodes on the map by clicking. All nodes are simulated to have WLAN and a longer range wireless serial connection in a master-slave configuration (i.e. they can only communicate with the control station). The code then computes the connection matrix for the given network and computes the bandwidth optimized routes for a transmission from node 1 to node 3 and to all other nodes, respectively. For comparison reasons the necessary computational times are measured and printed to the shell. The generated network topology is also plotted, Fig. 12 is an example for such a network topology plot.

A.1 main.m

The following listing contains the main MATLAB function called to simulate the graph routing part of a DBGR routing agent.

Listing A.1: `main.m`

```
%% MATLAB graph stuff

clear all, close all, clc;

BKG                 = '../McKennaMotion/AerialPictures_McKenna/
    McKenna_MOUT_iNSpect_550x310_washedOut.png';
SCALE               = 61/200;     % pixel/feet - scale factor of the
    picture
EFFECTIVE_RANGE     = 600*SCALE; % radio range in feet x SCALE
FT2M                = 0.3048;     % scale factor meter/feet

MASTER_NODES        = [1];



%% flip BKG for display in normal axis
bkg= imread(BKG);
```

```
bkg = flipdim(bkg,1);

%% create a node list
%Here we create the list of nodes, mimcing the 'initialisation' of the
%nodes in the network. The node list should be equal to the inital data
%each node has about the rest of the network

nodelist = NodeList(bkg);

%% create a graph
%Here we create the weights for the edges of the graph. This mimics the
%routine the nodes perform themselves in ordet to obtain a valid graph
%matrix. All information necessary to perform this action should be
%collected from the nodes list create before.
disp('Creating network...');tic;

[connectionMatrix, WLAN, Serial] = ...
    CreateConnectionMatrix(nodelist, MASTER_NODES);
graph = biograph(connectionMatrix);


toc;

%% draw connectivity graph
%Visualize the connectivitMatrix
disp('Visualizing network...');tic;

ConnectionMatrix3D(:,:,1) = WLAN;
ConnectionMatrix3D(:,:,2) = Serial;
drawConnectivityGraph(bkg,nodelist, ConnectionMatrix3D);

toc;

%% Calculate Routes

disp('Single route...');
[dist,path]=GetOptimalPath(graph,1,3);


disp('All routes...');
[dist_all,path_all]=GetOptimalPath(graph,1);
```

A.2 NodeList.m

This function provides the interactive node placement on the background map (BKG) and returns a list of nodes comparable to a DLS location table.

Listing A.2: `NodeList.m`

```
function nodelist = NodeList (bkg_image);

MAX_TR_RANGE_W = 250; % maximal transmission range using W-LAN
```

```
MAX_TR_RANGE_S = 400; % maximal trnasmission range using wireless serial

%% Creating Notes per Clicking
figure('Name','Node_Placement');
axis;
hold on;
subimage(bkg_image);
% for p = 1:length(wp)
%     plot(wp{p}(:,1),wp{p}(:,2),PLOTSTYLE(p,:));
% end
but = 1; n=0;
while but == 1
    [xi,yi,but] = ginput(1);
    plot(xi,yi,'ro')
    n = n+1;
    pos(:,n) = [xi;yi];
end
hold off;
% text = sprintf('Pixel \n(~ %.2f m/px)',FT2M/SCALE);
% xlabel(text);ylabel(text);
clear text;

%% Creating the output
nodelist = [pos' ones(n,1)*[MAX_TR_RANGE_W MAX_TR_RANGE_S] ];
```

A.3 CreateConnectionMatrix.m

This function takes the node list and creates the reachability and connection matrix. As an added feature to the proposed creation of the cost, this implementation tries to mimic the 802.11b bandwidth downscaling by first calculating the physical distance between two nodes and dependent on that determines the actual reachable bandwidth (utilizing `WLAN_SCALING`). In order to make the identification of a already found connection easier, an *edge list* stores all assumed to be possible connections.

Listing A.3: `CreateConnectionMatrix.m`

```
function [connectionMatrix, M_WLAN, M_Serial] = CreateConnectionMatrix(
    nodelist, master_node_list)

BANDWIDTH    = [11e3 38.4]; % kbit/s, high to low, according to the modes
EDGE_WEIGHT = [10 1];
WLAN_SCALING = [   0,    1;    % 11 Mbit/s
                 100, 0.5;    % 5.5 Mbit/s
                 200, 2/11;   % 2 Mbit/s
                 250, 1/11]; % 1 Mbit/s

%% Preprocessing for the connection matrices
% Creating the reachability matrices for the different modes (WLAN,
    Serial)
% and creating a list of all the edges. Based upon the list of edges the
```

```
% weights for these edges are computed and a connection matrix for each
% mode is generated.

nn = size(nodelist,1); % number of nodes

for mode=1:2 % selecting a mode: WLAN or Serial
    range=2+mode; % getting the max range for this mode
    Matrix = NaN(nn,nn);
    edge = [];
    for row=1:nn
        PosSrc = nodelist(row,:);
        for col=1:nn
            if row == col
                Matrix(row, col) = 0; % distance to itself
            else
                PosDst = nodelist(col,:);
                dist = norm(PosDst-PosSrc);

                % check if this node is reachable
                if dist < nodelist(nn,range)
                    % if we are in serial we need to check wether the
                        node
                    % is a master
                    if mode == 2 && all(master_node_list - row)
                        Matrix(row,col) = Inf;
                    else
                        Matrix(row,col) = dist;
                        % add another edge to the list of detectec ones
                        % this is needed for the weighting of the edges
                        % later on. Also add defaults for throughput
                        % format:[src, dst, dist, throughput];

                       switch mode
                            case 1 % WLAN 802.11b
                                r = 1;
                                while dist >= WLAN_SCALING(r,1)
                                    throughput = WLAN_SCALING(r,2);
                                    r = r+1;
                                end
                                clear c;
                            case 2 % serial
                             throughput = 1;
                            otherwise
                                error('Illegal Mode');
                        end

                        new_edge = [row, col, dist, throughput];
                        edge = [edge;new_edge];

                    end
                else
                    Matrix(row,col) = Inf;
                end
            end
        end
    end
```

```
%% Processing the edge list to calculate the weights for the edges
% In order to find an 'optimized' path using graph theory, the weights of
% the edges are of utmost importance.
% The serial link is assumed to be a point to point connection with propper
% means in place to minimize interference. Hence ALL serial connections are
% treated equally (with full bandwidth). This results in the same weight on
% all edges in the serial mode.
% For wireless we create a litte more sophisticated model. Based upon the
% limitation of the spectrum we degrade the bandwidth availabe for each
% connection/edge based upon how many other connections the particular node
% possibly has. I.e. an edge that is the sole connection to one node has
% more availabe bandwidth than one out of fife that connect to one specific
% node. On top of that one technically has to consider the fallback levels
% of the WLAN connection. 802.11b operates at 11Mbit/s only in the optimal
% case. It then scales down to 1Mbit/s for lossy/long distance links.

% Gather the connection-per-node information and update the conn@src and
% conn@dst entries

    conn_at_src = zeros(nn,1);
    conn_at_dst = zeros(nn,1);

    for row=1:size(edge,1)
        conn_at_src(edge(row,1)) = conn_at_src(edge(row,1)) + 1;
        conn_at_dst(edge(row,2)) = conn_at_dst(edge(row,2)) + 1;
    end

    % append to the edge list
    temp = edge;
    edge = [];
    for row=1:size(temp,1)
        density = [conn_at_src(temp(row,1)),...
                   conn_at_dst(temp(row,2))];
        throughput = temp(row,4);
        bandwidth = throughput*BANDWIDTH(mode)/mean(density);

        % calculating the weight for this edge:
        weight = GetScaledEdgeWeight(bandwidth,...
            max(BANDWIDTH),min(BANDWIDTH),...
            max(EDGE_WEIGHT), min(EDGE_WEIGHT));

        edge(row,:) = [temp(row,:) density weight];
    end

    % alter the initial connectivity matrix to reflect the new edge weight
    for row=1:size(edge,1)
        Matrix(edge(row,1),edge(row,2)) = edge(row,7);
```

```
        end

        switch mode
            case 1
                M_WLAN = Matrix;
                E_WLAN = edge;
            case 2
                M_Serial = Matrix;
                E_Serial = edge;
            otherwise
                error('Not a valid mode for connection matrix');
        end

    end

    %% Stitching the matrices to get one graph
    % In order to get a global graph we create three layers: Node, Node_W,
        and
    % Node_S. The node layer is the actual node. Nodes can send and receive.
    % The layers Node_W and Node_S reprensent the different wireless media.
    % Hence each node is connected to its media layers. Only nodes in the
        same
    % media layer can be connected.
    % This results in a matrix as follows (for two nodes and two media
        layers):
    %
    %
    %      ||  1  | 1w  | 1s  ||  2  | 2w  | 2s
    %  ====||=====|=====|=====||=====|=====|====
    %  1   ||  0  |  1  |  1  || inf | inf | inf
    %  ----||-----|-----|-----||-----|-----|----
    %  1w  ||  1  |  0  | inf || inf |  X  | inf
    %  ----||-----|-----|-----||-----|-----|----
    %  1s  ||  1  | inf |  0  || inf | inf |  X
    %  ====||=====|=====|=====||=====|=====|====
    %  2   || inf | inf | inf ||  0  |  1  |  1
    %  ----||-----|-----|-----||-----|-----|----
    %  2w  || inf |  X  | inf ||  1  |  0  | inf
    %  ----||-----|-----|-----||-----|-----|----
    %  2s  || inf | inf |  X  ||  1  | inf |  0

    connectionMatrix = Inf(3*nn,3*nn);
    BigDiag = [0 1 1;1 0 inf;1 inf 0];

    for row=1:nn
        for col=1:nn
            if row == col
                row_range = (row-1)*3+(1:3);
                col_range = (col-1)*3+(1:3);
                connectionMatrix(row_range,col_range)=BigDiag;
            else
                row_range = (row-1)*3+(2:3);
                col_range = (col-1)*3+(2:3);
                connectionMatrix(row_range,col_range)=...
                    [M_WLAN(row,col) inf; inf M_Serial(row,col)];
```

```
        end
    end
end
```

A.4 GetScaledEdgeWeight.m

This function gets called from `CreateConnectionMatrix.m` and performs the exponential scaling described in Sec. 6.3.2.3.

Listing A.4: `GetScaledEdgeWeight.m`

```
function weight = GetScaledEdgeWeight(bandwidth,...
    MAX_BANDWIDTH,...
    MIN_BANDWIDTH,...
    MAX_WEIGHT,...
    MIN_WEIGHT)
% Approximate the scaling with an exponetial function
% y = e^-[a(x-b)]

% MAX_BANDWIDTH = 11e3;
% MIN_BANDWIDTH = 38.4;
%
% MAX_WEIGHT = 10;
% MIN_WEIGHT = 1.00000000001;

if MIN_WEIGHT == 1
    MIN_WEIGHT = MIN_WEIGHT + 1e-4;
end

LY = log(MAX_WEIGHT)/log(MIN_WEIGHT);
b = (LY*MAX_BANDWIDTH - MIN_BANDWIDTH)/(LY-1);
a = log(MAX_WEIGHT)/(b-MIN_BANDWIDTH);

weight = exp(-a*(bandwidth-b));
```

A.5 GetOptimalPath.m

This function utilizes the MATLAB build in Dijkstra algorithm (`shortestpath`) in order to compute the optimal/shortest path through the graph. The function also measures the runtime of its execution from `tic;` to `toc`.

Listing A.5: `GetOptimalPath.m`

```
function [dist, path_hr, pred_hr] = GetOptimalPath(graph, src_hr, dst_hr
    )
% This function takes the node-IDs and transferst them into the
% corresponding IDs in the connectionMatrix. These IDs are then put into
% the MATLAB function shortestpath. The results are then transferred
    back
% into readable node-IDs.
```

```
%
%            1     2     3      4     5     6
%        ||  1  | 1w  | 1s  ||  2  |  2w | 2s
%     ===||=====|=====|=====||=====|=====|====
% 1   1  ||  0  |  1  |  1  ||  inf |  inf |  inf
%     ---||-----|-----|-----||-----|-----|----
% 2   1w ||  1  |  0  | inf ||  inf |  X  |  inf
%     ---||-----|-----|-----||-----|-----|----
% 3   1s ||  1  | inf |  0  ||  inf |  inf |  X
%     ===||=====|=====|=====||=====|=====|====
% 4   2  ||  inf | inf | inf ||  0  |  1  |  1
%     ---||-----|-----|-----||-----|-----|----
% 5   2w ||  inf |  X  | inf ||  1  |  0  |  inf
%     ---||-----|-----|-----||-----|-----|----
% 6   2s ||  inf | inf |  X  ||  1  | inf |  0
tic;
MODES = 2; % how many network modes are there? (i.e. the graph has MODES
    +1 layers)


% convert the human readable (_hr) IDs to graph IDs
src = 1+(src_hr-1)*(MODES+1);


% call shortestpath
if nargin >=3
    dst = 1+(dst_hr-1)*(MODES+1);
    [dist, path, pred] = shortestpath(graph, src, dst);
elseif nargin == 2
    [dist, path, pred] = shortestpath(graph, src);
else
    error('Input arguments do not match...');
end



% convert path and pred into human readable (_hr) form, where X is the
% node, X1 the interface for mode 1, X2 for mode 2, etc...

if iscell(path)
    for j=1:length(path)
        % Since we are dealing with the combined graph here, a lot of
            the
        % nodes are not of interest to us. They represent the actual
        % hardware (i.e. the WLAN or serial module). Hence we can
            eliminate
        % all paths leading to nodes, that are not actual nodes. (i.e.
        % eliminate all paths to hardware instead of the node.)

        id=path{j};
        length_id = length(id);

        if mod(id(length_id)+MODES,MODES+1) == 0
            for i=1:length_id
                id_hr(i)=10*ceil(id(i)/(MODES+1))+mod(id(i)+MODES,MODES
                    +1);
            end
```

```
            path_hr{j}=id_hr;
        else
            path_hr{j}=[];
            dist(j) = NaN;
        end
    end
else
    id = path;
    for i=1:length(id)
        id_hr(i)=10*ceil(id(i)/(MODES+1))+mod(id(i)+MODES,MODES+1);
    end
    path_hr=id_hr;
end

id = pred;
for i=1:length(id)
    id_hr(i)=10*ceil(id(i)/(MODES+1))+mod(id(i)+MODES,MODES+1);
end
pred_hr=id_hr;
toc
```

REFERENCES

[1] ACM, "Mobicom - the annual international conference on mobile computing and networking.". Available via: `http://www.sigmobile.org/mobicom/`, cited Dec. 2007.

[2] ACM, "Mobihoc - the acm international symposium on mobile ad hoc networking and computing.". Available via: `http://www.sigmobile.org/mobihoc/`, cited Dec. 2007.

[3] ACM, "Sensys - the acm conference on embedded networked sensor systems.". Available via: `http://sensys.acm.org/`, cited Dec. 2007.

[4] ACM, "Vanet 2007 - the fourth acm international workshop on vehicular ad hoc networks.". Available via: `http://www.sigmobile.org/workshops/vanet2007/`, cited Dec. 2007.

[5] ACM SIGMOBILE, "Mobisys - the international conference on mobile systems, applications, and services.". Available via: `http://www.sigmobile.org/mobisys/`, cited Dec. 2007.

[6] Banerjee, S. and Misra, A., "Minimum energy paths for reliable communication in multi-hop wireless networks," in *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, (New York, NY, USA), pp. 146–156, ACM, 2002. DOI:10.1145/513800.513818.

[7] Barriac, G., Mudumbai, R., and Madhow, U., "Distributed beamforming for information transfer in sensor networks," in *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, (New York, NY, USA), pp. 81–88, ACM, 2004. DOI:10.1145/984622.984635.

[8] Boleng, J., *Exploiting Location Information and Enabling Adaptive Mobile Ad Hoc Network Protocols*. PhD thesis, Colorado School of Mines, 2001.

[9] Brown, A. S. and Carter, D., "Geolocation of unmanned aerial vehicles in gps-degraded environments," 26 - 29 September 2005 2005. AIAA-2005-7011.

[10] Camp, T., Boleng, J., Williams, B., Wilcox, L., and Navidi, W., "Performance comparison of two location based routing protocols for ad hoc networks," in *Proceedings of IEEE INFOCOM*, pp. 1678–1687, 2002.

[11] Christmann, C. and Johnson, E. N., "Design and implementation of a self-configuring ad-hoc network for unmanned aerial systems," in *Proceedings on Infotech@Aerospace 2007 Conference and Exhibit*, (Rohnert Park, California, United States), AIAA, 2007. AIAA-2007-2779.

[12] Füssler, H., Mauve, M., Hartenstein, H., Käsemann, M., and Vollmer, D., "A comparison of routing strategies for vehicular ad hoc networks," Tech. Rep. TR-2002-003, Department for Mathematics and Computer Science, University of Mannheim, 09.12.2004 2002. Available via: `urn:nbn:de:bsz:180-madoc-7404`.

[13] (http://nsnam.isi.edu/nsnam/index.php), "The network simulator - ns-2.". Available via: http://nsnam.isi.edu/nsnam/index.php, cited Dec. 2007.

[14] Hu, Y.-C., Perrig, A., and Johnson, D. B., "Ariadne: a secure on-demand routing protocol for ad hoc networks," in *Proceedings on the 8th annual international conference on Mobile computing and networking*, (Atlanta, Georgia, USA), pp. 12–23, ACM Press, 2002. 570648, ACM 1-58113-486-X/02/0009. DOI:10.1145/570645.570648.

[15] JAUS WG, "Joint architecture for unmanned systems.". Available via: http://www.jauswg.org, cited Dec. 2007.

[16] Jenkins, A., Henkel, D., and Brown, T., "Sensor data collection through unmanned aircraft gateways," in *Proceedings on Infotech@Aerospace 2007 Conference and Exhibit*, (Rohnert Park, California, United States), AIAA, 2007. AIAA-2007-2747. Available via: http://pecolab.colorado.edu/papers/jenkins-AIAAsdc.pdf.

[17] Johnson, D. B. and Maltz, D. A., "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing* (Imielinski and Korth, eds.), vol. 353, Kluwer Academic Publishers, 1996.

[18] Karp, B. and Kung, H. T., "Gpsr: greedy perimeter stateless routing for wireless networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 243–254, ACM, 2000. DOI:10.1145/345910.345953.

[19] Ko, Y.-B. and Vaidya, N. H., "Location-aided routing (lar) in mobile ad hoc networks," in *Mobile Computing and Networking*, pp. 66–75, 1998.

[20] Koller, A., "Design, implementation, and testing of a multi-uav communication architecture," 4/29/2005 2005. Unpublished.

[21] Kong, J. and Hong, X., "Anodr: anonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, (New York, NY, USA), pp. 291–302, ACM, 2003. DOI:10.1145/778415.778449.

[22] Kurkowski, S., Camp, T., Mushell, N., and Colagrosso, M., "A visualization and analysis tool for ns-2 wireless simulations: inspect," in *MASCOTS '05: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, (Washington, DC, USA), pp. 503–506, IEEE Computer Society, 2005. Available via: http://toilers.mines.edu/pub/Public/PublicationList/MCS-06-01.pdf, DOI:10.1109/MASCOT.2005.11.

[23] Li, J., Jannotti, J., De Couto, D. S. J., Karger, D. R., and Morris, R., "A scalable location service for geographic ad hoc routing," in *Proceedings on the 6th annual ACM international conference on Mobile computing and networking*, (Boston, Massachusetts, United States), pp. 120–130, ACM Press, 2000. DOI:10.1145/345910.345931.

[24] Nahm, K., Helmy, A., and Kuo, C. C. J., "Tcp over multihop 802.11 networks: issues and performance enhancement," in *Proceedings on the 6th ACM*

international symposium on Mobile ad hoc networking and computing, (Urbana-Champaign, IL, USA), pp. 277–287, ACM Press, New York, NY, USA, 2005. 1062725. DOI:10.1145/1062689.1062725.

[25] NIST, "Kernel aodv.". Available via: `http://w3.antd.nist.gov/wctg/aodv_kernel/`, cited Dec. 2007.

[26] Paquereau, L. and Helvik, B. E., "A module-based wireless node for ns-2," in *WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator*, (New York, NY, USA), p. 4, ACM, 2006. Available via: `http://www.q2s.ntnu.no/~paquerea/ns.html`, DOI:10.1145/1190455.1190457.

[27] Park, V. D. and Corson, M. S., "A highly adaptive distributed routing algorithm for mobile wireless networks," vol. 3, pp. 1405–1413 vol.3, 1997. DOI:10.1109/INFCOM.1997.631180.

[28] Perkins, C., "Ad hoc on demand distance vector (aodv) routing," 1997.

[29] Perkins, C. E. and Bhagwat, P., "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, (New York, NY, USA), pp. 234–244, ACM, 1994. DOI:10.1145/190314.190336.

[30] Perkins, C. E., Royer, E. M., Das, S. R., and Marina, M. K., "Performance comparison of two on-demand routing protocols for ad hoc networks," *Personal Communications, IEEE [see also IEEE Wireless Communications]*, vol. 8, no. 1, pp. 16–28, 2001. DOI:10.1109/98.904895.

[31] Ramakrishnan, K., "An improved model for the dynamic routing effect algorithm for mobility protocol," 2004. Available via: `http://etd.uwaterloo.ca/etd/kramakri2005.pdf`.

[32] Ramanathan, R. and Redi, J., "A brief overview of ad hoc networks: challenges and directions," *Communications Magazine, IEEE*, vol. 40, no. 5, pp. 20–22, 2002. DOI:10.1109/MCOM.2002.1006968.

[33] Römer, K., "Time synchronization in ad hoc networks," in *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, (New York, NY, USA), pp. 173–182, ACM, 2001. DOI:10.1145/501436.501440.

[34] SISC, "Ieee standard for modeling and simulation [m and s] high level architecture [hla] - federate interface specification," *IEEE Std 1516.1-2000*, pp. i–467, 2001.

[35] Stefano, B., Imrich, C., Violet, R. S., and Barry, A. W., "A distance routing effect algorithm for mobility (dream)," in *Proceedings on the 4th annual ACM/IEEE international conference on Mobile computing and networking*, (Dallas, Texas, USA), pp. 76–84, ACM Press; New York, NY, USA, 1998. DOI:10.1145/288235.288254.

[36] Tayal, A. P. and Patnaik, L. M., "An address assignment for the automatic configuration of mobile ad hoc networks," *Personal Ubiquitous Comput.*, vol. 8, no. 1, pp. 47–54, 2004. DOI:10.1007/s00779-003-0256-5.

[37] TOILERS Reserach Group, "Efforts on manet routing protocols/services.". Available via: http://toilers.mines.edu/Public/CodeList, cited Dec. 2007.

[38] TOILERS Reserach Group, "inspect.". Available via: http://toilers.mines.edu/Public/NsInspect, cited Dec. 2007.

[39] Uppsala University, "Aodv-uu.". Available via: http://core.it.uu.se/core/index.php/AODV-UU, cited Dec. 2007.

[40] Uppsala University, "Dsr-uu.". Available via: http://core.it.uu.se/core/index.php/DSR-UU, cited Dec. 2007.

[41] Wetherall, D., "Mit object tcl.". Available via: http://otcl-tclcl.sourceforge.net/otcl/, cited Dec. 2007.

[42] Wikipedia, "Ad-hoc protocol list.". Available via: http://en.wikipedia.org/wiki/Ad_hoc_routing_protocol_list, cited Dec. 2007.

[43] Wikipedia, "Ieee 802.11.". Available via: http://en.wikipedia.org/wiki/IEEE_802.11, cited Dec. 2007.

[44] Wikipedia, "Ieee 802.11 rts/cts.". Available via: http://en.wikipedia.org/wiki/IEEE_802.11_RTS/CTS, cited Dec. 2007.

[45] Wikipedia, "Piconet.". Available via: http://en.wikipedia.org/wiki/Piconet, cited Dec. 2007.

[46] Yi, S., Pei, Y., Kalyanaraman, S., and Azimi-Sadjadi, B., "How is the capacity of ad hoc networks improved with directional antennas?," *Wirel. Netw.*, vol. 13, no. 5, pp. 635–648, 2007. DOI:10.1007/s11276-006-8147-0.